Resource orchestration for virtualized mobile networks

Gouareb, Racha

Awarding institution:
King's College London

Resource Orchestration for Virtualized Mobile Networks

Racha Gouareb

Centre for Telecommunications Research
School of Natural and Mathematical Sciences

King's College London

A thesis submitted to King's College London
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

February 23, 2020

Abstract

The anticipated growth in cloud computing leads to looking for a new way to design, deploy, and manage network services.

In this thesis, the problem of VNFs placement and routing across the physical hosts are studied to minimize overall latency. The latency is defined as the queuing delay within both, the edge clouds and the network links. In that respect, latency optimization takes a holistic view by considering not only VNFs chaining and placement problem but also considering the flows routing aspect. These two problems are inter-related and have a major impact on the network latency.

The variation of the network traffic affects the network performance by increasing the response time of the control plan when it is overloaded, which raises the issue of reliability and scalability. Multi-controller SDN-based networks calls for approaching the problem of controllers assignment as a multi-objective problem, by assigning switches to controllers in order to balance the load between the controllers. Furthermore, in this thesis we have considered traffic load migration with respect to computing resources.

To this end, two main problems are studied in this thesis: the placement and routing of Virtual Network functions, and controller assignment problem. We have considered different methods to measure a number of performance metrics such as delay and resource utilization. The analysis of simulation results and performance evaluations are presented to show the effectiveness of the proposed models in terms of latency and load balancing.

Acknowledgements

The completion of this thesis could not have been possible without the support and assistance of so many individuals. I sincerely appreciate their support, and I am very thankful to every one who helped me and supported me throughout this exciting and challenging journey. I am very grateful to my supervisor, Dr. Vasilis Frederikos, for his guidance, helpful advice, patience, and immense knowledge he showed throughout my PhD. His critiques and advice helped me to gain the knowledge and confidence I needed and to overcome times of difficulties. Thank you for your support and helpful suggestions, I will be forever thankful to you. A special thank to Prof. Hamid Aghvami, who was always present to give his thoughtful guidance and support. Whenever needed, he showed immense flexibility and understanding and found the perfect words to motivate me and pushed me to go over my limits. As a mentor, he is the first person who taught me the importance of hard work and patience. Without his support, I would not accomplish the work presented in this thesis. I wish to thank my family for their love and encouragement. I want to thank my aunt and mother, Zineb, who gave me all the love and support I need, she helped me to reach my objectives and tough me how to be a strong and grateful women. I am grateful to my parents, Fawzia and Abdelilah who unconditionally supported my dreams and helped me to be the strong person I am today, and my sister Roya, who always encourages me and believes in me. Last but not least, I would like to thank my husband Alexis for supporting me, always pushing me to work harder and believe in myself. I would like to dedicate my work to the people who mean so much to me. To my father who is no longer in this world, but his memories continue to inspire my days. Unfortunately, I cannot thank everyone by name because it would take a lifetime. To my colleagues and friends at King's College London in Center of Telecommunications Research, I

would like to thank them all for providing such a pleasant and friendly environment to work.

Contents

List of Figures

List of Tables

Glossary

3GPP Third Generation Partnership Project.

APPA Advanced Predictive Placement Algorithm.

BER Bit Error Rate.

BFD Best Fit Decreasing.

CAPEX Capital Expenditure.

CN Core Network.

CPU Central Processing Unit.

DPI Deep Packet Inspector.

EIGRP Enhanced Interior Gateway Protocol.

eMBB Enhanced mobile broadband.

EPC Evolved Packet Core.

ETSI European Telecommunications Standards Institute.

FCFS First Come First Served.

FDD Frequency Division Duplex.

FFD First Fit Decreasing.

GA Genetic Algorithm.

HMFMR Heuristic Multi Features Multi Requests.

HSFMR Heuristic Single Feature Multi Requests.

HSFSR Heuristic Single Feature Single Request.

IaaS Infrastructure as a Service.

IDS Intrusion Detection System.

ILP Integer Linear Programming.

IoT Internet of things.

KVM Kernel-based Virtual Machine.

LSA Link State Advertisement.

LSDB Link State DataBase.

LTE Long Term Evolution.

MANO Management And Orchestration.

MFMR Multi Features Multi Requests.

MILP Mixed Integer Linear Programmming.

MIP Mixed Integer Programming.

MME Mobility Management Entity.

mMTC Massive machine type communications.

MTC Machine-Type Communications.

NFV Network Function Virtualization.

NFVI Network Function Virtualization Infrastructure.

NLP Non-linear Optimization Problem.

NP-hard Non-deterministic Polynomial-time hard.

NR New Radio.

OFDM Orthogonal frequency-division multiplexing.

OPEX Operational Expenditure.

OSPF Open Shortest Path First.

PaaS Platform as a Service.

pps Packet per Second.

QAM quadrature amplitude modulation.

QoS Quality Of Service.

QP Quadratic Programming.

RAM Random Access Memory.

RAN Radio Access Network.

RIP Routing Information Protocols.

SaaS Software as a Service.

SBCs Session Border Controller.

SDN Software Defined Networking.

SFC Service Function Chain.

SFMR Single Feature Multi Requests.

SFSR Single Feature Single Requests.

TDD Time Division Duplex.

URLLC Ultra-reliable and low-latency communications.

vEPC Virtualized Evolved Packet Core.

VIM Virtualized Infrastructure Manager.

VM Virtual Machine.

VMs Virtual Machines.

VNF Virtual Network Function.

VNFM VNF Manager.

VNFs Virtual Network Functions.

VPN Virtual Private Network.

Chapter 1

Introduction

In recent decades, communication networks have been witnessing a massive growth in data traffic that needs a considerable processing capacity and communication as well as to support a vast number of users, in addition to the growth of user requests with more variety of customized requirements (such as low latency and high capacity).

The utilization of network, storage, and compute resources, the maintenance of hardware appliances and marketing new services have a noticeable effect on both Operational Expenditure (OPEX) and Capital Expenditure (CAPEX). As the consumption of different resources such as network and storage and the number of network services grows, the cost of deployment, management, and maintenance of hardware resources increases. Additionally, some resources can be unused or not fully utilized, while others are overused. Virtualization enables the optimization of computing and networking resources dynamically. In a virtualized infrastructure, it is easier to migrate virtual network functions (VNFs) dynamically based on the requirements. Network Functions Virtualization (NFV) enables functionalities to be virtualized and shared between different users/tenants, by decoupling network functions from physical infrastructure [1]. Therefore, by simplifying the configuration and installation of new services, virtualization enables service providers saving time (one third of the testing time of traditional networks) [2] and money.

To satisfy the user demands, the 5th generation of mobile networks is expected to deliver a new design to support specific functionalities. Updating existing network functionalities and re-designing networks require introducing new hardware equipment and significant network changes which increase service providers expenses.

The next network generation is expected to support a big range of applications such

as tactile internet, remote surgery, human to machine and machine-to-machine communications. Additionally, 5G is expected to overcome the high cost of integrating and upgrading new network functionalities by deploying new network architectures [3]. Therefore, 5G network faces a large number of requirements such as ultra-low latency and high reliability requirements, enhanced mobility management, and high flexibility in addition to lowering CAPEX and OPEX.

One of the most important supported application by 5G is Internet of things (IoT), which can be defined as a network of a large number of connected objects, combining different Machine-Type Communications (MTC). In IoT, objects are remotely controlled and managed, and a considerable amount of data is collected, exchanged and analyzed.

Therefore, IoT is aiming to collect, process and analyze a large amount of data. Resource utilization is a primary challenge, where the virtualization plays a major role by supporting sharing, reusing and managing hardware resources [4]. Hence, virtualization and softwarization play a significant role in the management of resources and maximization of resource utilization to support IoT applications.

To keep up efficiently the diversity of the 5G use cases, new concepts such as network slicing are required. Network slicing can efficiently support diversity of requirements of use cases by running various applications in different network slices (virtual networks). One slice is a specific collection of network functions and resource allocation modules. Every slice is isolated from each others to protect themselves from any cyber attack.

New services and use cases have different requirements, such as ultra-reliable and low-latency communications (URLLC) that requires high reliability, security, and demands very low network latency. Additionally, New technologies allow diverse industries to develop new models and deploy various logical networks over one infrastructure. Two complementary but independent concepts; SDN and NFV are introduced to meet the stringent requirements of 5G different use cases, Enhanced mobile broadband (eMBB), URLLC and Massive machine-type communications (mMTC). With the modernization of the networks, the increase in traffic loads and the dependency of businesses to their networks, Software Defined Networking (SDN) and NFV allow flexibility and automation. In fact, virtualization is essential to upgrade new network functions and locate them where and when needed in the network.

SDN enables the configuration and programmability of forwarding rules, by abstracting network control from forwarding functions in addition to improving networks agility to meet the changes dynamically. As a complementary paradigm and an abstraction of network functions, NFV uses virtualization technology. By decoupling network functions from one or more physical network infrastructure(s) to run in software such as virtual machines instead of hardware, network functions can be easily moved through different locations, installed and upgraded in any location of the network.

The European Telecommunications Standards Institute (ETSI) describes a hight-level NFV framework composed of three principal elements; VNFs, NFV Infrastracture (NFVI) and NFV Management and orchestration (MANO). In NFV, the virtualization layer shares physical resources between VNFs including computing, storage and network to create virtualized resources that can be deployed, managed and executed over one or different virtual machines (VMs) [1]. On the top of the NFVI, VNFs are software packages deployable in one or different VMs allowing scalability, reusability and speed.

The NFV management and orchestration is responsible for the management and orchestration of all resources in the cloud data center. MANO have many challenges such as instantiation, routing, placement, and chaining of VNFs. Many studies on NFV have attempted for diverse objectives in various scenarios. Latency is one aspect of performance that needs to be emphasized, especially for time-sensitive traffic such as haptic, audio, and video. To deploy a wide range of services, improve network flexibility, and facilitate the deployment and maintenance of network services, virtualization of network functions promises to make the networks more agile and efficient. NFV enables network service providers to deal with the increase of data traffic and the deployment of new services by offering a more scalable and flexible network. For better control of the network, SDN complements NFV by decoupling the network control from the data forwarding through programming. Therefore, combined SDN and NFV gives more flexibility in network control and encourage innovation to improve network performance.

SDN encompasses different technologies such as programmable networking, network automation and network virtualization, and is adopted by different network operators and large companies such as Google. SDN enables administrators to update when necessary any switch rules and prioritize the flexibility in managing traffic loads and contributes in designing new technologies. SDN controllers determine all forwarding

components behavior in the network. While southbound APIs, such as Open flow protocol, allow the communication between the control plane and data plane and deliver information to network elements. Northbound APIs enable the communication between controllers and applications, which helps networking programmability. Therefore, SDN provides flexibility in control which enables the control of many devices at the same time and a network of applications. In order to capture new opportunities and face the explosion of new services, businesses need to reduce time to market, adapt to network changes faster, and maintain quality of service (QoS).

1.1 Motivation

We are witnessing a paradigm shift from conventional networks to virtualized networks. Service providers/operators have turned their interests towards NFV. From implementing and deploying network functionalities in hardware to running multiple services, NFV allows organizations to deploy services as software in VMs to offer the required services. NFV enables the market to reduce the cost of equipment and time to test and evaluate new services as well as allowing faster marketing. To ensure a service experience using softwarization to run VNFs with respect to customer demands; placement, and chaining of VNFs should be optimized to ensure a better Network performance.

Within the network functions virtualization ecosystem, high availability and low latency are two service quality benefits. Service providers have high expectations for 5G and the network functions virtualization to make delay-critical services such as remote surgery, a reality. Therefore, network services should be placed, chained, and routed through the network considering users/tenants stringent QoS and service-level agreement requirements. Routing and placement of virtual networks as well as SDN controller assignment to network elements play an essential role in improving network performance and the overall network cost. Hence, it is a big challenge to find the optimal placement, especially in large scale topology networks.

Many challenges have to be addressed, and different questions should be answered in order to enable the use of NFV and SDN in 5G. One of the main challenges to be investigated, VNFs placement and deployment, performance evaluation and provisioning services, considering different network metrics. NFV challenges can be investigated and solved independently. Additionally, the integration with known concepts such as

SDN should be considered. Therefore, investigating in softwarization will have a real added value to the new emerging technologies such as 5G and IoT as they are getting more and more attention.

With the growth of traffic flows, the policy execution of service chains is hard to maintain manually. Therefore, SDN offers more flexible and programmable environment. In software defined networks, the controller can be considered as a critical network element for the overall performance of the network. Furthermore, the spatio-temporal variation of network traffic affects network performance. It can increase the response time of the control plane when it is overloaded, raising the issues of reliability and scalability. The combination of service placement and chaining with the aid of SDN improves network flexibility. Furthermore, controllers are designed as the core and brain of SDN. By centralizing the controller, we are bringing more flexibility to the network management, but in the case of failure, the whole system can be affected. This thesis proposes a multiple-controller solution to achieve balance in traffic load and avoid single point of failure compared to a single-controller solution.

1.2 Contributions

This thesis aims at dealing with two important and related problems, it addresses the first problem of VNFs placement and routing and the second problem of controller assignments.

In order to manage and orchestrate the virtual network functions throughout the network and avoid limiting the network performance or capacity (e.g., Bottleneck), we have focused on placing and routing VNFs by:

- Presenting mixed integer linear programmming (MILP) problem, considering not only the delay generated by serving user/tenant demands but also the cost of routing using a bi-objective optimization problem.

  Considering the two costs helps faster execution time of service requests.

- Optimizing the distribution and utilization of available resources and meeting user/tenant requirements. The objective has been to minimize the overall accumulated latency on both: the edge clouds, where VNFs are running and the flow routing paths.

- Delay optimization leads to maximization of service requests. Also, we have considered other factors such as load balancing through virtual links.
- For larger networks, to accelerate the process of routing and placement, we have aimed at using a heuristic method in order to consider a larger number of NFV features.
- To optimize latency and solve placement and routing problem, we have considered and compared different approaches detailed in the following chapters, with respect to the delay requirement and network cost.

The main objective of the solution presented in Chapter 3 and Chapter 4 is to enable operators to increase the acceptance rate of strict delay requirement requests and avoid capacity restrictions.
To counter the problem of load balancing in the control plane, we have formulated and solved in Chapter 5 the problem of controller placement considering traffic load and traffic load migration as:

- Mixed integer quadratic programming problem, where we have assigned switches to controllers considering their loads and with respect to the computing resources, with the main objective of load balancing among the controllers.
- Linear model as a mixed-integer linear programming problem minimizing the maximum load to be applied to a more realistic scenario, and performance evaluations of the proposed models on real-world network topology.

This thesis key objective is to allow a more significant number of users to utilize the network resources efficiently in terms of capacity and time and accept more delay-critical service requests.

The work publications listed below presents the contributions of this thesis.

Published Articles:

[5]. R. Gouareb, V. Friderikos, A.-H. Aghvami, "Virtual network functions routing and placement for edge cloud latency minimization", IEEE J. Sel. Areas Commun., vol. 36, no. 10, pp. 2346-2357, Oct. 2018.

[6]. R. Gouareb, V. Friderikos, A. H. Aghvami, "Delay sensitive virtual network function placement and routing", Proc. Int. Conf. Telecommun. (ICT), pp. 394-398, Jun. 2018.

[7]. R. Gouareb, V. Friderikos, A. H. Aghvami, "Placement and Routing of VNFs for Horizontal Scaling", Proc. Int. Conf. Telecommun. (ICT), April. 2019.

[8]. R. Gouareb, V. Friderikos, A. H. Aghvami, M. Tatipamula "Joint Reactive and Proactive SDN Controller Assignment for Load Balancing", Proc. Int. IEEE Global Telecommunications Conference Workshops, 2019. GlobeCom Workshops 2019.

1.3 Structure of the Thesis

The rest of the thesis will be organized into five Chapters, presented as follows:

Chapter 2 gives an overview of 5G and some concepts related to the work presented in this thesis. The chapter discusses the relevant previous work for VNFs placement and routing, in addition to load balancing among SDN controllers and covering the background of the topic. In Chapter 3, we define the problem of VNFs placement and routing for delay-sensitive applications considering the delay within links. Chapter 4 describes a new approach to VNFs placement and routing. In this chapter, we have considered not only link delay but also inter-cloud delay, and we have extended the work to solve the problem in both cases: vertical and horizontal scalings. Load balancing for SDN controllers is studied in Chapter 5. In this chapter, we have examined the problem of the SDN controller assignment for load balancing.

We have concluded the thesis in the last chapter with some remarks, discussions of interesting points and challenges to be considered for the current work extension and future work.

Chapter 2

Background Study

2.1 Introduction

In this chapter, we present a detailed literature review, where we discuss recent developments on the topic from previous related research work. Additionally, we give an overview of the background, and we define general concepts related to the problem outlined in this thesis.

The first part of this chapter introduces 5G networks describing the New Radio(NR) and highlighting one of the essential concepts considered to build the 5th generation core network. In the second part, we cover NFV technology, architecture, and different components. Additionally, we present the main challenges and approaches studied in recent research works, especially the one related to VNFs placement and routing.

The third part gives an overview of SDN controller load balancing, discussing the background and different approachs to load balancing and SDN controllers assignment problem. Then, We provide an overview of routing protocols and queuing models. In the last section of this chapter, we present different ways of classifying and solving optimization problems, and we cover some heuristics and greedy algorithms.

2.2 5G Networks

5G is a huge project aiming to make the world even more connected than what it is today, with lower energy consumption and maintenance cost. 5G supports next-generation technologies and services such as autonomous driving, smart cities and

factories, and augmented reality [9]. Therefore, the existing mobile architecture is no longer sufficient for the low latency requirements and different use cases, which pose several challenges to operators. 5G makes it possible to support the coexistence of human-centric and machine to machine and machine to human applications [10].

The Third Generation Partnership Project (3GPP) has defined three categories to classify 5G services:

- EMBB which have high bandwidth requirements for applications such as augmented reality and high-resolution video streaming.
- mMTC requires massive connectivity, better energy efficiency, and from low to high data rate to serve a large number of machine-type devices (e.g., sensors). The added value expected from IoT to 5G is its ability to support massive amounts of simultaneous connections.
- URLLC requires high reliability and very low latency for latency-sensitive services. Low latency is also seen as a must-have for the deployment of innovations (e.g., autonomous driving).

2.2.1 5G New Radio

Different requirements are needed to support new services in 5G radio such as low latency, high reliability, and massive connectivity. To improve mobile network performance, flexibility, scalability, and efficiency, 5G New Radio(NR) is a new interface developed for 5G to support a large number of different services, deployments, and diverse 5G device-types. 5G NR uses OFDM, a multi-carrier modulation method using technologies such as massive MIMO and scalable subcarrier spacing to offer higher available bandwidth and different spectrum bands. 5G is designed to offer faster data rates, reduce latency , 4ms for mobile use and 1ms for devices such as automated cars. To overcome LTE traffic congestion issues, 5G uses of a higher frequency bands which makes great amounts of bandwidth available.

3GPP introduced 5G NR Non-standalone (NSA) mode standardization in April 2016 and its commercial release is expected before 2020 [11]. NR key benefits are:

- Ultra-lean design (minimum “always-on” transmissions)
- Advanced multi-antenna techniques (multi-beam antenna)

- Forward compatibility (flexible in adopting new techniques)
- Wide spectrum range (operating in frequencies below 6 GHz and above 20 GHz)
- Low latency

NR operates in both Frequency Division Duplex (FDD) and Time Division Duplex (TDD) with flexible frame structures. Orthogonal frequency-division multiplexing (OFDM) is used for both uplink and downlink to facilitate device-to-device communications. Optimum waveforms for OFDM are proposed for uplink and downlink transmissions with variable cyclic prefix [12]. NR supports variable bandwidth (RF channel) and subcarrier. Below 6 GHz frequency band, subcarriers spacing of 15, 30, 60 kHz are used for carrier bandwidths of 50, 100, and 200 MHz, respectively. For above 20 GHz frequency bands subcarriers of 60 and 200 kHz are used for bandwidths of 200 and 400 MHz, respectively.

In addition to long term evolution (LTE) modulation schemes (such as 16, 64, 256 QAM), a new modulation scheme, $\pi/2$-BPSK is introduced for the low data rate uplink transmission (IoT) for NR. $\pi/2$-BPSK has the same Bit Error Rate (BER) performance as that of BPSK with less envelope variation. Therefore, lower peak-to-average power ratio (PAPR) which can be transmitted through a high power-efficient non-linear UE power amplifier with less distortion. NR standardizes an advanced low-density parity-check (LDPC) coding. The reasons for this choice are its reduced coding latency and complexity, block length flexibility and supporting lower data rates than LTE turbo codes.

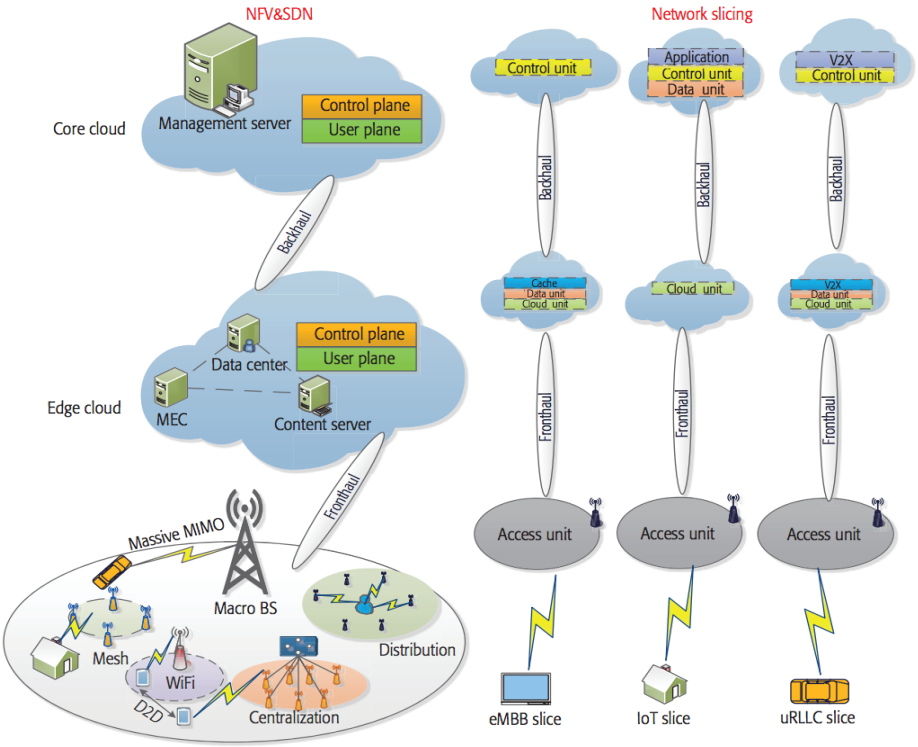

Figure 2.1: Network slicing based 5G system architecture

2.2.2 5G Core Network

As described by the 3GPP, 5G system architecture is composed of one or more access networks and a core network. In the first phase of 5G deployment, LTE core network, Evolved Packet Core (EPC) is used. In the second stage, the 5G infrastructure is expected to be based on the network slicing concept. The core network is composed of network functions and supports data connectivity and new deployment technologies such as NFV and SDN that enables network slicing as illustrated in Figure 2.1.

The core cloud represents a centralized architecture of the Core Network (CN) transformed into a core cloud. It provides functionalities of the CN such as mobility management and aims to reduce control signaling and delays of data transmissions. On the other hand, the edge cloud is a centralized pool of radio access network (RAN) virtualized functionalities that provide data forwarding and control plane functions such as baseband processing [13].

Network slicing allows efficient use of core network resources (bandwidth, computing power, and storage). As a collection of network functions and resource allocation

modules, network slices are isolated from each other and share one common physical infrastructure. Every network slice can be considered as a virtual network with customized functionalities an on-demand service to address vertical industries requirements and QoS demands.

Each application can be supported by one of the slices, and each slice can be optimized for that particular application requirements. In the unit of network slicing management, as shown in Figure 2.2, SDN controller operates and controls the entire virtual network by connecting the data layer and vertical applications. The virtualized network function manager (VNFM) is responsible for mapping physical network functions to VMs, while Virtualized infrastructure manager (VIM) is responsible for the allocation of virtualized resources. As the core part of the network slicing manager, the network management and orchestration unit is the core part of slicing management, because it is responsible for creating, activating, and deleting network slices.

As the next step for 5G commercialization, it will be allowing dual connectivity of both 4G and 5G access to the new 5G core deploying NFV and SDN, which enable the implementation of network slicing. However, the fifth generation is still facing different challenges, such as isolation requirements among slices to achieve the required security level.

eMBB, mMTC (IoT) and URLLC use cases have big potential but business models are still essential to justify the huge investment required for 5G virtualized infrastructure. Additionally, the involvement of other vertical business sectors to share capital expenses is essential for the deployment of shared physical infrastructure.

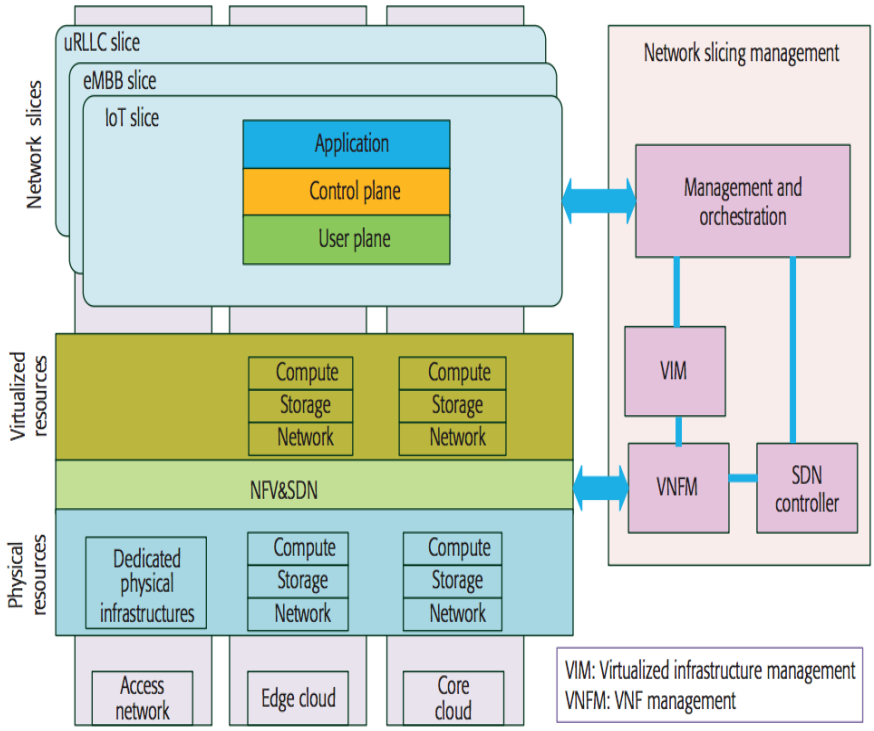

Figure 2.2: Network slicing management

2.3 NFV and SDN

In order to process multiple operations simultaneously and balance the load among servers, moving toward data-centric models allow to reduce traffic congestion episodes and move closer to the end users. Therefore, network function virtualization gives users/tenants the ability to place and deploy network functions on the cloud.
To separate control and data, two complementary but independent concepts are introduced: SDN and NFV have been studied during this last decade [14], [15]. SDN decouples the control and data planes and enables programming the behavior of the network using well-defined interfaces. As a complementary paradigm, NFV uses virtualization technology to run network functions on software that can be easily moved through different network locations, which reduces CAPEX, OPEX, space and power consumption.

2.3.1 Network Function Virtualization

ETSI defined NFV as a network architecture, proposed by a research group NFV ISG composed of twenty largest Telecommunications Operators such as British Telecom in a white paper published in October 2012 [16]. NFV architecture, as shown in Figure 2.3, can be divided into three main components, NFVI that provides essential resources for the data plan, MANO corresponds to the control layer and VNF to the application layer. VNFs are offered by vendors to perform a specific network function such as firewall, router, and load-balancer, where their combination can compose a Network slice with one or different software programs that run on shared hardware to create different touch points[17].

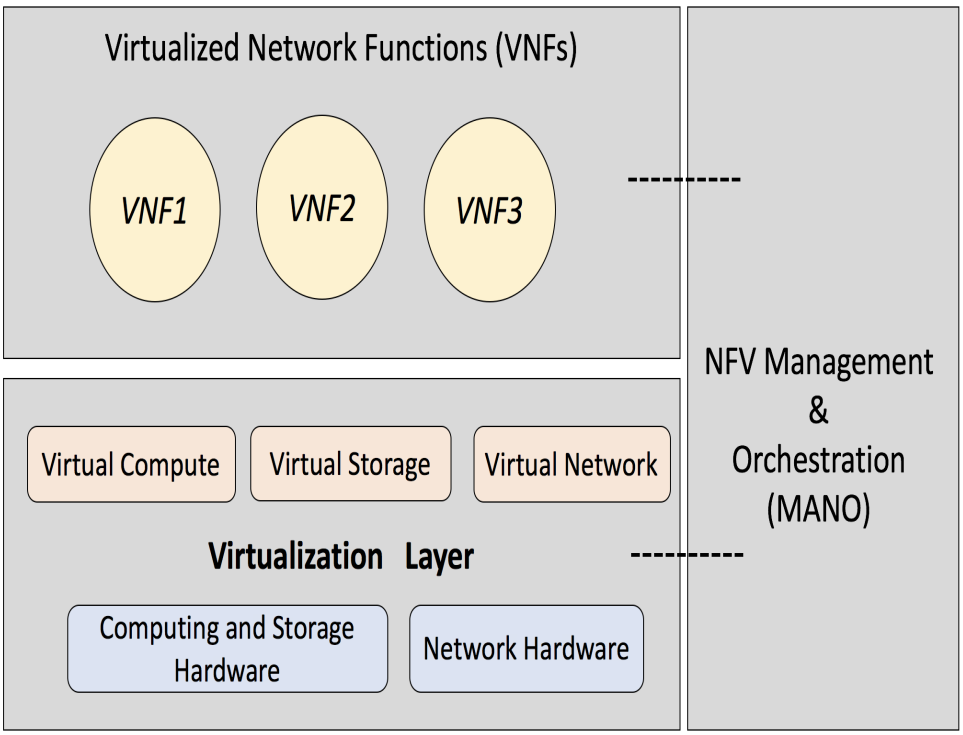

Figure 2.3: High-Level ETSI NFV framework

2.3.1.1 NFV Architecture

As the Network evolution goes from a hardware-centric to a software-based approach network, a new framework is needed to support NFV technology. The standardized architecture of the new NFV framework should allow multi-vendors to be able to implement VNFs, configure and manage their life cycles. With three key blocks,

ETSI defines the relationship between VNFs, their management, and their interdependencies. ETSI defines the first block of NFVI as a multi-tenant infrastructure with the hardware, software, and the virtual components needed to host VNFs. Virtual machines replace network nodes in the traditional architecture, by building software on the top of virtual machines from NFVI to virtualize Network functions independently to hardware. MANO is a separate block communicating with both NFVI and VNF blocks and responsible for the orchestration and life-cycle management of physical/virtual resources and VNFs in addition to their allocation.

In more details, the first layer of NFVI architecture with a physical layer composed of compute, and storage hardware called compute nodes for core processor and servers such as Tower server and Rack server. Storage nodes are considered as storage devices for temporary or permanent storage. Network hardware can be an industry switch such as IBM RackSwitch [18] and allow communication between network elements through a physical connection.

In the lower block, NFVI framework, a hypervisor or container constitute the virtualization layer which is based between the physical and virtual infrastructure. This layer creates an isolated environment for different tenants and use physical resources to instantiate or remove a VM/container.

On top of the virtualization layer, virtual compute, virtual storage, and virtual network composes the last part of NFVI. Virtual compute, storage, and network are representing respectively the virtualization of hardware processing such as Central Processing Unit (CPU), storage hardware which creates resource pools and interconnection between virtual components such as VMs.

The second block of NFV framework represents abstracted functionalities from physical nodes. VNF layer creates isolated instances of network functions. Also, the VNFs life cycle is managed by the Element Management System (EMS) and VNFMs by sharing VNFs information. VNFs does not exist in VNF layer only; we can find different network functions in NFVI layer or MANO block such as controller functions. VNFs instances can be implemented and managed by either VMs or containers as shown in Figure 2.4.

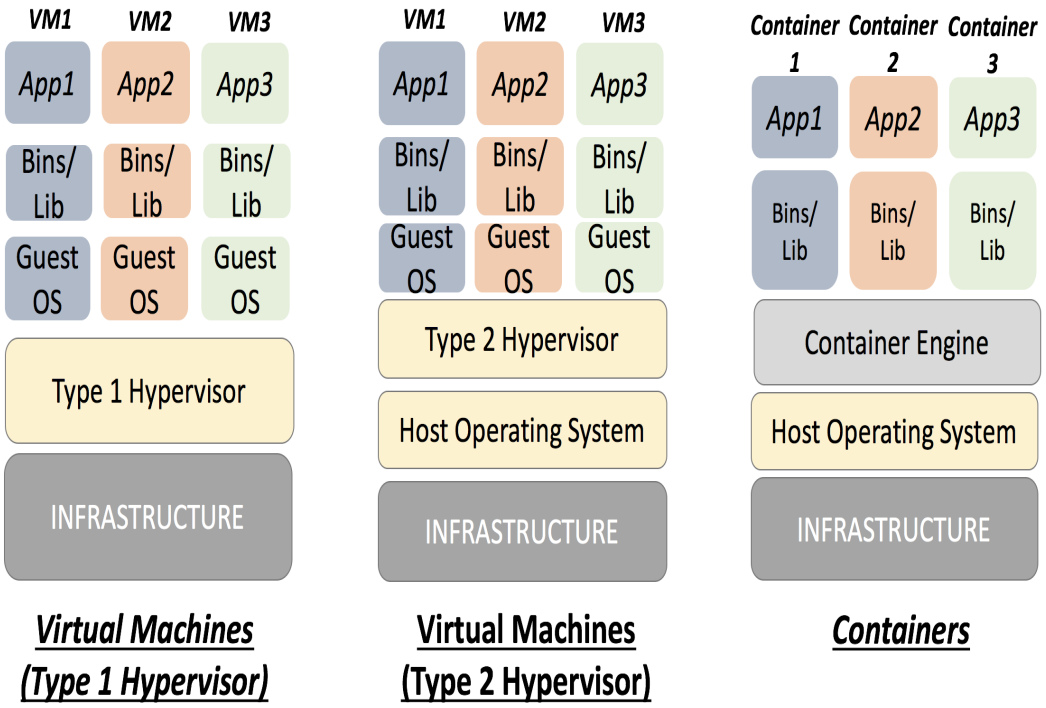

Figure 2.4: VM vs. container

A VM is a software running on the top of the physical platform, where hypervisors are between the hostOS and GuestOS to instantiate and run VMs.

Each VM has its own kernel, which means that VMs can host different applications and still run on the same physical hardware. Therefore, VMs can provide better security than containers. However, VMs are heavyweight which add some complexity to the processing of every step of their lifecycle. The size of VMs can affect their portability and make the migration from a machine to another one complicated. We can find different types of hypervisors, type 1 and type 2 that are two ways to run virtual machines. Type 1 hypervisors such as KVM (Kernel-based Virtual Machine) and Microsoft HyperV are defined as a bare-metal hypervisor that runs on physical hardware.

The hypervisor, installed on bare metal hardware is a lightweight operating system compared to type 2 hypervisors that are installed on top of a standard operating system.

Type 2 hypervisors known as hosted hypervisors such as Oracle VirtualBox and VMware Player [19] enable us to run a different and isolated operating system within

the main one.

On the other hand, as a lightweight version of VM, containers come with better resource utilization, faster and easier deployment, and versioning. Different containers can be sharing the same OS kernel, they are very light (megabytes in size), more portable compared to VMs and the start-up time from 40 sec for a VM to 50 ms for a container. However, by sharing the same OS kernel, security is one of the biggest challenges for the use of containers as they are more vulnerable to attacks than VMs. Containers have various benefits such as size and performance but face different problems related to security, such as protection from malicious host and applications [20].

The last block of NFV architecture is NFV management and orchestration layer; according to ETSI the role of MANO can be defined through the roles of VIM, NFV Orchestrator (NFVO) and VNFM. VIM is responsible for managing network resources. The NFVO determines the optimal path to chain different VNFs; NFVO is also capable of creating and orchestrating VNFs. The role of VNFM is the management of VNFs where each VNF can have only one VNFM, but one VNFM can be responsible for different VNFs.

2.3.1.2 Virtual Network Function Chains

We define a user/tenant request as VNFs chain, where demands and VNFs chains are used interchangeably. By defining a request, we describe the requirements for the service, including the required network functions and the performance. A service chain consists of a set of linked network functions, which are provided by specific VNFs. Each request requires the execution of virtualized network functions allocated on one or multiple VMs. We can create a service chain using one or multiple VNF instances that results in different connected services with different characteristics.

2.3.1.3 Virtual Network Function Placement

In this section, we introduce the optimization problem of VNFs placement with different objectives such as minimizing resource utilization, e.g., [21] and [22] or maximizing the network throughouput such as in [23]. Different approaches measure different metrics such as end-to-end latency, power consumption, number of active nodes, and network traffic.

NFV offers a more flexible network environment, where it is easier to move functionalities and place them; which have a significant role in network optimization.

In [24], Defang Li et al. tackle the problem of VNF placement in edge computing, considering the trade-off between node and link capacities. The problem formulation comes into an Integer Linear Programming (ILP) model to optimize resource consumption in terms of node capacities and bandwidth. A polynomial-time priority-based greedy is proposed for a large scale version of the problem. To evaluate the performance of their solution, the authors compared results to a Random Greedy and a heuristic solution proposed in [25] based on reusing VNFs and delay minimization.

Aris Leivadeas et al. propose in [26] an interesting approach to service function chain (SFC) placement and deployment problem in cloud-based network architecture. The main objective is to minimize end-to-end communication delay with their minimum deployment costs by placing VNFs chains in both cloud and edge infrastructure. The evaluation of the proposed model considers three topology cases for a more realistic environment. Four delays have been defined to calculate the end-to-end delay and model the problem: propagation delay, transmission delay, processing delay, and queuing delay. For large scale scenario, a Tabu search algorithm is proposed based on local search and it is composed of five stages. The last step of the search algorithm is the termination criterion defined as the number of iterations needed for the algorithm to converge to a local optimal solution. The number of iterations was fixed to 1400 as the algorithm converges to an optimal solution. The evaluation of the approach performance has considered measuring the utilization of resources (number of servers, links, and function instances) and the overall delay. It is shown that the MIP results outperform Tabu search algorithm, which gives a closer result to the optimal solution than algorithms based on resource usage only.

A different approach to VNFs placement in [27] presents the model as a MILP problem. In addition to a heuristic BC based on the calculation of betweenness centrality(number of shortest paths from a node s to a node t through a node v over the number of all shortest paths from a node s to a node t) to ensure the placement of critical VNFs in the node with minimum communication delay. In [27], Hassan Hawilo et al. consider minimizing the intra-communication delay between VNF instances. Additionally, the authors are capturing all carrier-grade requirements of applications such as performance, scalability, and QoS and assuming different dependencies between VNFs. As a result, the evaluation of the proposed MILP and BACON heuristic

algorithm showed a better performance compared to a greedy-based k-shortest path algorithm from the state of the art.

In [28], M Savi et al. present the impact of processing resource sharing between different VNFs on service function chains placement. For this, two costs are studied : The context switching cost defined as the cost from loading CPU process context to enable the execution of multiple CPU processes sharing the same CPU. That means sharing multiple cores by different processes. Secondly, the upscaling cost defined as the cost from load balancing traffic among CPU cores, which has more latency and processing costs. The problem is modeled as an ILP problem, and a heuristic Cost-aware algorithm to increase the number of instances is described. The analysis of the results shows that the cost of context switching has more impact on VNFs placement, especially for low latency requirement services.

Oussama Soualah et al. present their work in [29] to optimize the utilization of resources, by placing and chaining the requests of VNF forwarding graph, considering sharing VNFs between tenants. They propose an ILP based algorithms to process requests in batch online using the minimum resource and processing a maximum number of requests. Therefore, they have measured different metrics during the simulation such as the execution time that shows the performance of the algorithm when scaling the problem. Additionally, they have considered the number of active nodes, the power needed to host service requests, service provider revenues, and the amount of rejected requests. The paper concludes that approaching the problem of requests placement and chaining as a batch enables service providers to define requests priorities. Additionally, they use ILP for solving bigger size problem since the complexity is controlled.

M. Tajiki et al. [30], propose an extension to there previous work [31] with different heuristic approaches considering various aspects of the problem. The authors, not only consider VNFs placement but also flow allocations and routing, without the objective of minimizing the overall delay. However, the authors propose an energy-aware approach along with other constraints on flow size, ordering of VNFs, and flow latency.

Different metrics are measured to evaluate the performance of the proposed algorithms, such as path length, link utilization, node utilization, and computational complexity. Also, a comparison between heuristic and ILP results has been presented, showing a maximum of 14% of optimality-gap.

2.3.2 Software Defined Networks

SDN is a new emerging technology and a complementary approach to virtualized solutions that helps organizations to cope with high-bandwidth requirement, dynamic applications and the transition from the traditional network to a virtual environment. Therefore, to keep up with network evolution and high demands, SDN separates the network into two segments: data plane and control plane that we detail in the following section. Traditional networks are based on the physical infrastructure that is composed of network elements such as switches and routers that can not communicate or run without the interaction with the physical nodes. On the contrary, by allowing resources to be provisioned from a central location, SDN as a software-based network runs on a NFV infrastructure and allows providers/users to leverage new devices and grow there network without the need to invest in more physical nodes. Therefore, providers/users have more flexibility and freedom in the management of network functionalities and network configuration from a centralized location.

2.3.2.1 SDN Structure and Architecture

SDN is defined as new network architecture, and its framework can change from an organization to another one. However, the general SDN framework is represented in Figure 2.5 [32] and is composed of three different layers defined as follows:

- SDN applications are the upper layer and include various types of applications from management, configuration, security to business software applications. This layer communicates with the control one to make decisions communicate behavior and request resources via APIs.
- Control plane is a layer with a global view of the network and an intelligent logic. SDN controller receives requests from the application layer and communicates an abstract view of the network to applications by interacting with the hardware devices and extracting needed information.
- Different Network devices such as routers and switches compose the data plane, responsible for forwarding data and managed by the control plane.

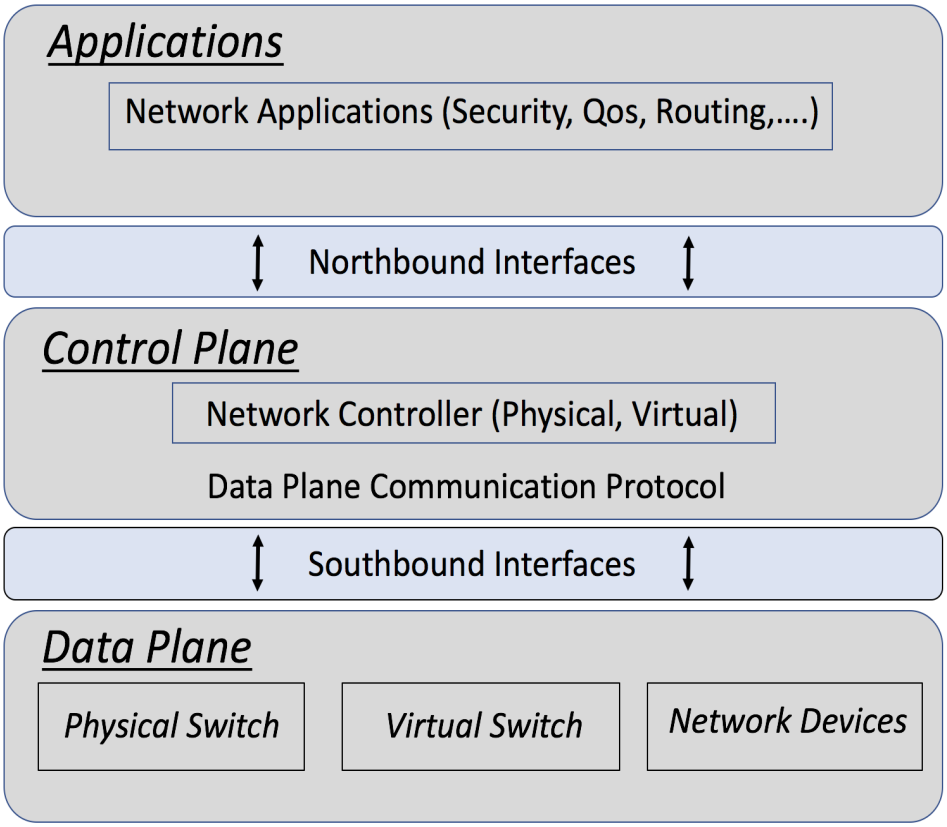

Figure 2.5: SDN architecture

2.3.2.2 SDN Multi-controllers

As the first design of SDN architecture, a single controller is managing all network switches. When one of the network components needs routing information, it should request the routing information for the new packet by sending a Packet-in to the controller. Considering a single controller architecture, might affect the processing time of requests within the network as well as the overall performance in the case of a failure.

Additionally, as request rate increases, requests traffic sent to the controller from different network elements increases. Therefore, the capacity of the controller might not allow the processing of all requests [33].

However, a new approach to SDN architecture is multi-controllers, where request traffic is distributed between controllers and controller placements is optimized. Multi controllers or multiple domains in SDN is to have different areas, each with a central controller managing a group of switches. Each controller has a partial vision of the

network but has to communicate with other controllers to share network information. This can solve the problem of single-point failure, by using its own controller when another controller is down and increase network performance, security and scalability. However, multi-controllers raise multiple challenges such as controller placement and assignment, controllers communication cost, and load balancing.

2.3.2.3 Controller Placement and Assignment

In this section, we present exciting works on controller placement and assignment.

Hadar Sufiev et al. present an approach to SDN controllers assignment for load balancing in [34]. The authors present a dynamic controller clustering in order to obtain a number of clusters, where load is balanced between clusters and between controllers. Two heuristics are used for initialization and replacement and an algorithm based on the k-center approach to define K masters. The first step consist of defining K master controllers; the second step assign controllers to predefined empty clusters that contain only the master controller. In the first heuristic, the assignment of controllers is based on the distances between them; each controller is assigned to the cluster with the nearest master. While in the second heuristic, it is based on the load of groups, it means that it assigns the most loaded controller to the underloaded cluster. The main objective of this paper is to have less load on the serving controller and respect the maximum distance between controllers, in addition to evaluating the performance of a dynamic clustering compared to a fixed one.

Authors in [35] present a solution to the problem of controller placement problem (CPP) based on their previous work, which is different from [34]. Victoria Huang et al. consider in addition to controller workload distribution the communication delay and the control plane utilization. For this, the authors have developed an algorithm based on the combination of genetic algorithm and gradient descent (GD) optimization method for fitness evaluation. In more details, for the assignment of controllers, a CPP solution is presented considering the response time and control plane utilization. For evaluation, different approaches are used for comparison such as random heuristic approach based on a random selection of controllers, capacity-based greedy approach based on selecting the controller with the highest capacity in each iteration, direct optimization approach and K-mean approach adopted in minimizing the average communication delay. The simulations show that the algorithm presented

results in a high control plane utilization and low response time compared to heuristics used in the state of art.

A new approach to SDN controller assignment is presented in [36] for a multi-controller to avoid any network failure or damage. The work motivation is clearly defined in the chain failure phenomenon when one controller failure causes other controller failures. Therefore, Tao Hu et al. has proposed a dynamic slave controller assignment to prevent any network crash. Different constrained on controller availability and capacity (idle capacity, reserved capacity, and maximum capacity) are defined for the problem formulation. The optimization solution to slave controller assignment considers different metrics such as latency between switches and controllers, and load balancing based on the processing of flow requests. Additionally, the solution is measured based on flow request rate and controller utilization. Besides, the authors have presented a heuristic algorithm to improve efficiency with three modules state detection to identify controllers failure. Efficient slave assignment is the module that makes a new slave controller assignment and role adjustment model defines new controller roles (slave, master). Through simulation, this solution proved that the probability of network failure could be reduced and latency can be decreased. However, results can be improved for a more realistic scenario within a large scale network.

A different solution to the controller placement problem is presented in [37], where authors have highlighted the importance of multi-controllers or multiple domains approach. Each controller has a partial vision on the network, communicate, co-operate with other controllers and have a group of switches to manage. Therefore the placement and assignment of controllers in different domains and to groups of switches should be defined in respects to various metrics such as load balancing and latency. Ahmad Jalili et al. have proposed a bi-objective optimization, minimizing the end-to-end flow set up time defined as the time we need to set up forwarding rules in switches. Furthermore, authors considered minimizing the inter-controller communication latency caused by the communication between controllers to share network parameters. Additionally, the authors designed an optimizer following best-worst multi-criteria decision-making and a heuristic algorithm to define controller assignment to switches defining the paths between switches and controllers.

The heuristic method considers hope count metric defined as the number of nodes within the route between a switch and a controller, propagation latency, link utilization, and path reliability (the best path between each placed controller and its

assigned switches). By considering two realistic network topologies from state of the art: Abilene and Internet2 from Internet Topology Zoo, the evaluation of the proposed approach confirms the efficiency of the model compared to previous works.

2.4 Routing

Service requests are composed of a sequence of VNFs chained in a specific order, interconnected between them by virtual links and defined by a source and a destination. Network traffic needs to go through a specific number of chained VNFs to process service requests before reaching the destination node. As the traffic has to traverse servers in different geographical locations, it is essential to determine the ideal routing path with respect to QoS [38]. Therefore, routing is the process of selecting the best route from a source to a destination, based on traffic and network topology, through which network traffic can be dispatched.

2.4.1 Network Routing Protocols

Network routing protocols define the path for two routers or nodes in the network to be able to communicate between them. Therefore, routing protocols by sharing information find and select preferred routes to forward data packets throughout the network, from a source node to a destination node.

Routing protocols have to collect information about network topology to take decisions dynamically about preferred paths to forward traffic through network nodes.

We can define two main categories of routing protocols: Distance vector protocols such as Routing Information Protocols (RIP) and Enhanced Interior Gateway Protocol (EIGRP), where each router should share its routing table with its neighbors. Routers send their full routing tables periodically to their neighbors, in case any change occurs in the routing tables. Therefore, the information is shared throughout the entire network, but this makes the process very slow. The routing decision is made based on the distance using algorithms such as Bellman-Ford algorithm, and the best route is the one with the fewer hops. RIP protocol was developed for small to medium networks, and the maximum number of hops is 15.

However, Link state protocols such as Open Shortest Path First (OSPF) are mainly used in scalable large IP networks [39], overcoming the restricted number of hops

in distance vector protocols. In this category, routers do not frequently share their routing table. Link state protocols exchange the topology information throughout the autonomous system. OSPF is one of the most popular link state protocol where routers exchange or request information using different types of packets such as Link State Advertisement (LSA) and Link State DataBase (LSDB).

LSA packets are exchanged between routers in the same area and carry all topology information and can have different functionalities. A link state update message is sent to answer a link state request message for a missing part or information of the shared database. Thus, LSDB is built, or a portion is updated, this to maintain the same link state database between all routers. By considering network link states, each router finds the shortest end-to-end path to every possible destination using shortest path algorithms such as Dijkstra algorithm to build their routing tables.

In this thesis, we use the Dijkstra Algorithm for routing, to find the shortest or lowest cost path between two points in the same graph. The algorithm goes through different steps [40] as defined in Algorithm 8 in Appendix A. First, we initialize all distances from a source to any vertex to infinite, the parent node (previous node) as undefined and the distance to the source is zero. As we fixed the starting node, the next step is to choose the neighbor node with the smallest weight/cost. Once we move to the selected node, we check all its neighbors and calculate for each of them the distance based on the weights of the links that lead to each neighbor from the starting node. The smallest cost will be defining the shortest distance for that node. Therefore, the output of Dijkstra is a set of shortest paths from the starting node to all reachable nodes in the graph.

2.4.2 Multipath Routing

Multipath routing is a routing technique for assigning different paths for the same source and destination nodes. It can be effectively used for the minimum delay and packet loss during congestion, to increase bandwidth, improve reliability and quality of service. In contrast to the single path approach, multipath routing can better utilize network bandwidth and balance network traffic. To define multiple paths, different shortest route algorithms have been used such as Dijkstra and Bellman-ford-Moore [41], where not only one shortest path is set, but k-shortest paths. In this thesis, we define 2-shortest paths per request in a weighted network shown in Figure 2.6. The weights assigned to links are positive, pre-defined by the service provider and can

represent the cost or distance between two nodes. Weights can also be defined as link bandwidth or as the necessary power for transmission. In contrast to Bellman-ford algorithm, Dijkstra algorithm aims to find the shortest path based on link costs or weights, where all edge costs belonging to a route should be positive from source to destination node in a defined graph.

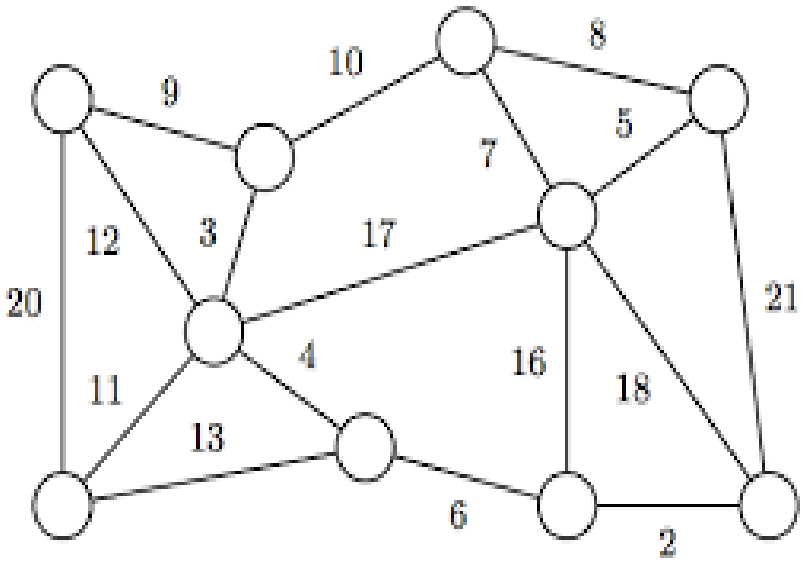

Figure 2.6: Weighted graph

2.5 Queuing Theory

Queuing theory is a mathematical methodology generally used to analyse the performance of a network. Minimizing delay is a major problem in the network, queuing models help to optimize routing and select routes with the minimum delay. At high traffic load, many data packets arrive at a node, the waiting queue for transmission build up, which might occur network congestion and bottleneck.

Network delay can be defined as the time for a data packet or a bit of data to go through the network from a source node to a destination node. The delay can be categorized in four different delays:

- Processing delay is the time to process a data packet header.
- Queuing delay is the time a packet spent in a queue.

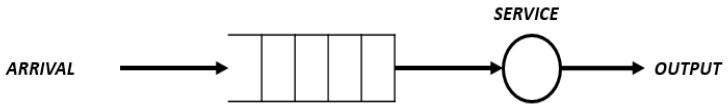

Figure 2.7: $M/M/1$ queuing model

- Transmission delay is the time needed to transmit all the bits of a packet through the link. The delay is defined by the packet size over the bit rate.
- Propagation delay is the time required for a bit to travel through the path and reach the destination. The delay is calculated by the distance over propagation velocity. Therefore, propagation delay is proportional to the path length.

In this thesis, we will focus on studying the queuing delay as it plays a major role in the overall network delay. Once a packet arrives at a node, it should be processed and transmitted. Every node and link have a limited capacity of processing and transmitting. If the packet arrival rates exceed the capacity of the node or link, the packets are pushed into the queue. Therefore, queuing delay can be defined as the time data packets spend in a queue, waiting to be processed or transmitted. Delay can be defined following the Little's Law based on the chosen queuing model. There are four commonly used queuing systems following Kendall's Notation $A/B/s$. A for Arrival distribution (M for Poisson, D for deterministic, and G for general), B for Service time distribution (M for exponential, D for deterministic, and G for general), s for the number of servers. Assuming that arrivals follow the Poisson distribution, queuing systems can be categorized as follows:

- $M/M/1$: Queue in a single server system and service time with an exponential distribution.
- $M/M/s$: Similarly to $M/M/1$ but in a multi server queuing system.
- $M/D/1$: Queue in a system with a single server and where service time are fixed.
- $M/G/1$: Queue modeled in a single server system where the job service time has an unknown, arbitrary distribution.

In this thesis, we assume that arrivals follow a Poisson distribution and that service times have an exponential distribution similar to [42][43]. We can model each link

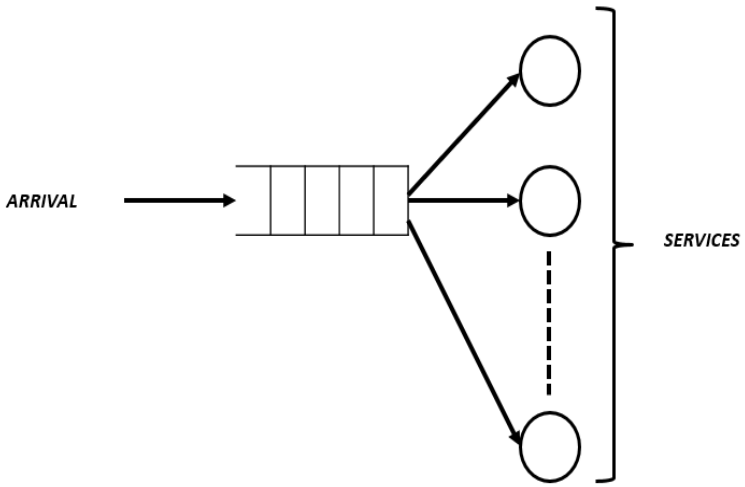

Figure 2.8: $M/M/s$ queuing model

and node/ VNF instance as an M/M/1 or M/M/m queue. Based on the queuing network model, the input process can be defined by packets of requests that arrive stochastically as a Poisson stream, where the arrival of each packet is independent of the others. Additionally, the packets arriving to the processing server(s) are either served or are waiting to be processed in the queue following a First Come First Served (FCFS) approach, this defines the queuing process. Also, each link, node or VNF instance handle each packet independently from the other. Packet Arrival rate is defined by λ and by assuming that packets have different sizes the service time is exponentially distributed.

Therefore, we used $M/M/1$ and $M/M/s$ queuing models as shown in Figure 2.7 and Figure 2.8 respectively.

2.6 Optimization

Optimization is a set of techniques that act to minimize or maximize one or different constrained or unconstrained functions to find the optimum, the best possible result. Analytic methods use different calculus to locate the optimal solution. As the problems are not identical and are under different circumstances, various ways can be used to calculate in a set of operations and find the optimal solution. An optimization problem such as optimal resources allocation, minimum processing time, shortest route .etc, can be expressed through three components: Objective function(s), decision variables and constraints.

Objective function defines the effort to minimize or the benefit to maximize and can be composed of one or multiple objectives. Multi objective functions are commonly used in the presence of a trade-off. In complex multi-objective functions, there is no single optimal solution for each objective function. Pareto optimal solution improves one objective function while degrades the other values. Therefore, Multi objective problems can be reformulated, either by using a weighted summation of functions or by keeping one function as the objective and putting other functions as constraints.

Variables are the second component of an optimization problem; decision variables are unspecified and can be discrete Boolean or continuous.

The third and final component is the constraints. Constraints are defined as the conditions to be satisfied to give the decision variables $x = (x_1, x_2, \ldots, x_n)$ the right values.

The general form of an optimization problem is defined by $f(x)$, the objective function in equation 2.1, $g_i(x)$ where $i = 1, 2, \ldots, m$ defines inequality constraints in equation 2.2 and $h_j(x)$ where $i = 1, 2, \ldots, p$ represents equality constraints in equation 2.3.

$$\underset{x}{\text{minimize}} \quad f(x) \tag{2.1}$$

$$\text{s.t.} \quad g_i(x) \leq 0, \quad i = 1, 2, \ldots, m. \tag{2.2}$$

$$h_j(x) = 0, \quad i = 1, 2, \ldots, p. \tag{2.3}$$

The first step in solving an optimization problem is to construct the mathematical model by defining different terms listed previously, objective function(s), variables, and constraints. In a second step, the type of a problem should be determined and this by defining in which category the issue falls.

Optimization problems can be classified differently. A problem can be unconstrained if there are no constraints on the defined problem and variables. On the other hand, a constrained problem might have one, two, or many equality or inequality constraints, which puts some conditions and boundaries on the set of possible solutions. Constraints can also be classified as linear, non-linear, or convex constraints.

The nature of the objective function plays a significant role in classifying the optimization problem and also defining the computational method to be used to solve the

problem. In the case where all the constraints are linear, the optimization problem is classified as a linear programming problem if the objective function is linear too. However, if the objective function is quadratic, then the problem is a quadratic programming problem. The problem is called non-linear if one of the objective functions or constraints include a non-linear term.

Depending upon the decision variable types, optimization problems can be classified as integer or real-valued model. The problem is an integer when some or all the variables can take only integer values. However, the real-valued model is when the set of decision variables can only take real values.

Problems can be defined as stochastic or probabilistic if the variables in the objective functions and constraints include random elements. The second type is deterministic models, where the set of conditions are defined. Therefore the results remain the same if the starting points do not vary.

The classification of the optimization problem helps to define a suitable method and technique to solve the problem. Unconstrained optimization aims to solve more general problems using different methods such as Direct search, Gradient, or Newton methods. We start by defining the Direct methods that do not need the knowledge of derivatives to search for an optimal point. This technique can be used for problems where the function is neither differentiable nor continuous.

However, the gradient method, such as gradient descent and the conjugate gradient use the knowledge of the first derivatives and are used for large scale problems. These methods consist of finding the local optimum based on the direction of the gradient of the function and the step size. Newton's methods are based on the knowledge of the first and the second derivatives. By using the second derivative, Newton's methods choose a more direct route, faster to reach a local maximum or minimum compared to the gradient descent.

In the case where both the objective function and the constraints are linear, we use linear programming methods. Every linear problem can be converted into a standard form before been solved. There are different ways to transform the form of a linear problem to a more standard form. If the problem is defined as a minimization problem, it can be converted to a standard maximization problem by multiplying the objective by (-1) and constraint of the form $a_i x_i \geq b_i$ can be changed to $-a_i x_i \leq -b_i$. Similarly, a non-restricted variable can be replaced by two restricted variables and

two inequality constraints. Therefore y_i can be transformed to $u_i - v_j$ by adding $u_i \geq 0$ and $u_i \geq 0$ as constraints.

In the case where the objective function or one or more constraints are non-linear functions, we refer to the problem as a non-linear optimization problem. Non-linear Optimization Programming (NLP) problems are the most general form of the optimization problem as all other defined problems can be considered as a subproblem of NLP problems.

By having a NLP problem, you have enough information to find a local minimum or maximum point. Whereas, it is challenging to know if there is other better local points or to define the global optimum [44]. Additionally, they might be different and discontiguous feasible regions and different starting points, which is time-consuming to go over all of them. Therefore solving NLP is complex. Accordingly, before trying to find the right solution algorithm, it is better to try to reduce the non-linear terms of the problem formulation to a linear mathematical model.

A simple reformulation might be able to solve the problem with a linear optimization method. As an example, non-linear terms can be approximated by piece-wise linear function [44] and the non-linear optimization problem can be transformed into a linear one. The approach of piece-wise linearization as used in this thesis, can be applied to a problem if the objective functions are separable functions, functions that can be written as a summation of functions for a single variable $f(x_1) + f(x_2) + \cdots + f(x_n)$ [45].

Therefore, piece-wise linearization can be the solution when the objective functions or the constraints or both are non-linear [46]. For this, a defined number of breakpoints will be denoted, and there corresponding linear function will be generated as shown in Figure 2.9. Giving the objective function as $f(x) = x^2$ the breaking point are 0, 1, 2 and 2.5 and there corresponding function values are 0, 1, 4 and 6.25. By applying the weighted sum, the piece-wise linear approximation of the function can be expressed as follows:

$$\lambda_1 f(x_1) + \lambda_2 f(x_2) + \lambda_3 f(x_3) + \lambda_4 f(x_4) = f(x) \tag{2.4}$$

$$\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 = x \tag{2.5}$$

$$\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1 \tag{2.6}$$

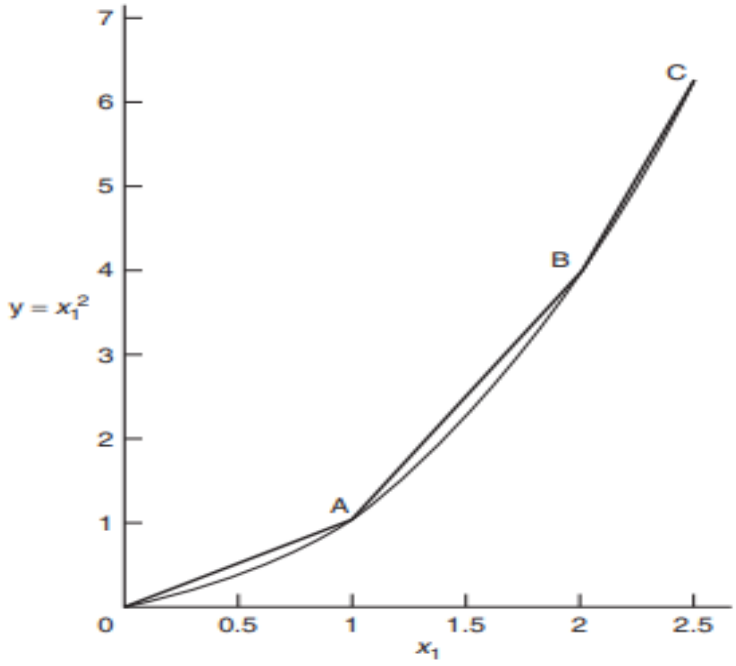

Figure 2.9: Piecewise linear approximation

Non-linear terms might be found in the form of products of variables. In that case, we can eliminate the product and replace it with a new variable and introduce new constraints to the problem model. The product of x_1x_2, if both binaries can be replaced by the new binary variable $y = x_1x_2$ and constraints (2.7),(2.8) and (2.9) are added.

$$y \leq x_1 \tag{2.7}$$

$$y \leq x_2 \tag{2.8}$$

$$y \geq x_1 + x_2 - 1 \tag{2.9}$$

Otherwise, if x_1 is binary and x_2 is continuous, the new variable y is continuous and constraints (2.10),(2.11) and (2.12)are added.

$$y \leq ux_1 \tag{2.10}$$

$$y \leq x_2 \tag{2.11}$$

$$y \geq x_2 - u(1 - x_1) \tag{2.12}$$

Different methods have been developed to solve the NLP problems, methods that are based on transforming the non-linear constrained problem into unconstrained sub-problems. These techniques are called exact methods such as Augmented Lagrangian method that uses a penalty term added to the objective function to eliminate constraints.

Other methods are based on solving a sequence of problems that converge to the solution of the original problem, and they are called sequential methods. Sequential Quadratic Programming method is known by solving large scale non-linearly constrained problems [47]. Sequential Quadratic Programming method uses the quadratic approximation to replace the objective function. A set of quadratic models with different resistance factors are solved. At every iteration, the resistance factor is reduced, and a quadratic problem is modeled, and its solution results in a new step toward the solution of the original non-linear problem.

Problems with a quadratic function and linear constraints are Quadratic Programming problems. Their general form can be defined in the objective function (2.13). Q is a $n \times n$-dimensional real symmetric matrix,c is a n-dimensional vector. Equality and inequality constraints are defined in (2.14) and (2.15) respectively, where A and Aeq are $m \times n$-dimensional real matrices and b and beq are m-dimensional real vectors.

$$\text{minimize} \quad \frac{1}{2}x^T Qx + c^T x \tag{2.13}$$

$$\text{s.t.} \quad A_x = b \tag{2.14}$$

$$Aeq_x \geq beq \tag{2.15}$$

Quadratic Programming (QP) problems can be considered as a particular case of nonlinear programming problems.

2.7 Heuristics

The cost of finding an optimal solution is not always affordable. When the size of a problem grows, it will take more time to find an optimal solution, especially for

real-world applications. Therefore, the execution time and space will increase.

NP-complete is a problem where no optimal solution is found, such as Traveling Salesman Problem [48]. Heuristic algorithms are used to find an approximate solution (sub-optimal) but not an accurate one as it does not guarantee the best solution. A heuristic algorithm might find the best solution, but it is very hard to be proven. A heuristic algorithm is designed to solve problems and make decisions in a fast manner by looking for a sub-optimal solution within the possible solutions. Besides, traditional search algorithm such as exhaustive and local search, divide and conquer, branch and bound are effective methods but face the time-complexity. Therefore, different modern heuristic algorithms [49] have been developed to overcome the time-complexity of the traditional ones such as:

- Simulated annealing algorithm
- Tabu search
- Artificial Neural Networks
- Genetic Algorithms that are based on mutation, crossover, and selection and are mainly used to generate a high-quality solution.

2.8 Greedy Algorithm

The greedy algorithm is a developed method that follows the problem-solving heuristic to find the local optimum. At each step, the algorithm is trying to find a way to an optimal global solution for the entire problem. The greedy algorithm does not guarantee the optimal solution, as it focuses on the information at the current stage only, not the entire problem. However, it does produce a local optimal solution, close to the global one with good run time. Five components should be defined to develop a greedy algorithm:

- A candidate set: Initial set to create a solution
- A selection function: Chooses candidates
- A feasible function: Decides if a candidate can be a part of the solution
- An objective function: Gives a value to a solution

- A solution function: Decides if the final solution is found

The greedy algorithm has been used for a wide range of problems such as routing problem where Dijkstra's shortest path algorithm is used. Some greedy algorithms do sometimes produce the global optimum solution, but most of the time, they yield to the local optimum solution only. However, such algorithmic techniques can be good to obtain an approximation of an optimization problem.

The greedy algorithm can be used in different classical cases and also as an approximation for non-deterministic polynomial-time hard (NP-hard) problems to overcome the polynomial time complexity of optimal techniques.

For applications in routing, Dijkstra algorithm uses a greedy algorithm to find the shortest routes in a directed graph with positive edge weights. Another application for greedy is to bin packing problem. This problem is defined by several packing bins with a specific capacity and a number of items of different sizes. The objective can be time efficiency, minimizing the number of bins, or balancing the distribution of items. For solving the problem of bin packing, different algorithms are used, such as next fit decreasing, first fit decreasing and best fit decreasing. The general form of best fit algorithm can be found in Algorithm 9 in Appendix A [50].

Clustering problems without the clusters knowledge are NP-hard, where the greedy algorithm such as k-means and k-centers can be used to partition the data-set. K-means clustering is used for data that does not belong to any category. By defining the number of clusters and a random selection of centroids, the objective of the algorithm is to assign a group to every element based on its distance to the centroid.

2.9 Summary

In this chapter, we have reviewed the main methods for optimization, routing and queuing, in the context of which we modeled the VNFs placement and routing problem and the controller assignments problem. For cases where we have a non-linear mathematical model, there exist techniques to reformulate the problem and methods to solve the model and find optimal or sub-optimal solutions.

Different approaches have been presented to solve the problem defined in this thesis. Therefore, we have been inspired by some of the previously published work, and we have discussed several approaches that have been proposed for similar or related

problems. In the next chapter, we present our approach to the problem of VNFs placement and routing.

Chapter 3

Delay Sensitive Virtual Network Function Placement and Routing

3.1 Introduction

NETWORK function virtualization (NFV) is a recent concept for sharing resources from one or more physical network infrastructures in order to improve flexibility of design, deployment and management of networking services. VNFs are hosted in virtual machines for the purpose of serving a user and/or a tenant request where the order of the requested functions are as important as the service itself. Therefore, the placement, chaining and routing of VNFs have a major impact on the overall networking cost which can be translated into financial cost, latency, resources utilization or other suitable metrics. In this chapter, we focus on the accumulated delay assuming multipath routing of flows and the assignment of service chains in virtual networks. We formulate the problem of placement and routing as a MILP problem, considering not only the delay cost generated by serving user and/or tenant demand, but also the cost of routing using a bi-objective optimization problem. The proposed solution aims to minimize the overall delay and the simulation results show a considerable gain in the performance and a better use of network resources compared to the conventional shortest path routing based solutions. The main contribution of this chapter is the formulation of network service chains placement using MILP model for the purpose of optimizing network latency and increasing the acceptance rate of strict delay requirements. This enables a larger number of users utilizing efficiently network resources in terms of capacity and time. The rest of the chapter is organized as follows. Section 3.2 reviews the related work. Section 3.3 describes the network

model, the mathematical formulation and states the load balancing problem in NFV as a mathematical formula. Discussion and evaluation results are given in Section 3.4. Section 6 concludes the chapter and highlights future work.

3.2 Background

Multiple studies on VNFs consider network latency as a critical attribute to insure QoS for costumers of both network and application providers. Such advanced features of service offerings has to take place via an automation and elasticity of resource distribution and allocation [51]. In this emerging environment, replacing network functions that are currently implemented on middle-boxes by VNFs, the issue of placement of VNFs and routing of network traffic are considered as important issues to be addressed to ensure efficient network operations. By considering aggregated data rates, the amount of used resources and latency in [52], independent graphs have been created using a heuristic method to optimize VNFs chaining and evaluate the effect of those decisions on all these three metrics.

A challenge that has been targeted in [53] related to finding a balance between performance and resource usage. To this end, three methods have been investigated: Round-robin heuristic for flexible placement solutions, Mixed Integer Programming method generating a set of solutions to the SFC and a queueing theory based method for latency estimation. The approach proposed in [53] is a SFC provisioning system (Stringer) able to minimize the infrastructures resources and the network delay without considering the routing cost.

An interesting approach is proposed in [54] solving the problem of joint service placement and traffic steering incrementally. The authors have formulated the VNFs placement and routing problem with the objective of minimizing both link and core resource utilization. For this purpose, authors have modelled the problem using mathematical programming aiming to provide efficient placement of service chains but considering latency as a constraint rather than an objective to minimize.

The work presented in [55] focuses in ensuring continuity and improving resilience of Service Chains (SCs) with a minimum number of resources and with respect to delay requirements. To achieve that, three integer linear programming models have been introduced for each type of failure. Again, in this work they have analysed the effect

of nodes capacity, failure type and the number of active nodes on network latency but no delay optimization objective is considered to solve the problem.

Some approaches such as [56], consider bit rate variation and dynamic bandwidth allocation to minimize scheduling delay and solve resource allocation problem for network services. Only two types of delays are considered: processing delay and transmission delay in order to serve more requests and cope with stringent service requirements.

The so-called FAST-RACE algorithm is proposed in [57] to reduce network congestion and increase network availability. The authors consider delay but focus on increasing the number of accepted demands by distributing the load through the network and avoiding overloaded link state. None of the above papers have considered the optimization problem of joint latency and routing cost taking into account queuing delay, multi-path routing and both link and node virtualization capacities which is the focus of this chapter.

3.3 Problem Description and System Model

As already eluded above, the optimal VNFs chaining and routing problem is an area that has gained significant research attention and the problem itself falls within the NP-hard optimization problems. In this section, we derive a compact formulation that linearizes the effect of delay to allow powerful linear integer mathematical solvers to be used to find optimal solutions. We consider the case where all VNFs of a service request are located at the same access/core network location, for example at the same edge cloud. Furthermore, we assume that all the service requests are already admitted in the network; in other words no admission control is considered in this work. Network flows should visit different network functions depending on the service such as for example, video optimizer, Deep Packet Inspector (DPI), Session Border Controller (SBCs) and Firewall in a specific order to be applied to the flow of data [52]. We define each service request as a chain of ordered functions represented as a directed graph similar to many research work such as in [58]. The nodes of the graph represent the network functions and the edges describe the link between the network functions. In our case, the problem reduces to the optimal routing and location of each service request. We define the required preliminary notations and parameters that will help us to formulate the mixed integer linear program.

3.3.1 Preliminaries

A mobile core/access network is modelled as an undirected graph $G = (K, L)$, where K denotes the set of nodes and L the set of links in the network. Let P be the set of paths in the network that can be composed of one or multiple links. F denotes the set of VNFs and f_i represents an instance of specific VNF$_i$. Each f_i consumes some physical resources (i.e., CPU cycles, fast/slow memory, etc.) if the function is implemented in a node. We aggregate the resource requirements for a request that might include more than one VNF (in a chain) as η_r, meanwhile, the amount of available resources of node k is denoted as another single column matrix Γ_k. Therefore, the assumption we made hereafter is that all VNFs are located on the same node with a chain being implemented internally in that node (such as for example an edge cloud).

We assume a convex function for packet forwarding delay on a link where C_l is the capacity of the link and x_l the total traffic in link l. By using the fundamental definition from the M/M/1 model in queuing theory, the delay function can be defined in equation (3.1).

$$D_l(C_l, x_l) = \frac{1}{C_l - x_l} \tag{3.1}$$

To linearize the delay profile of a link, so that a linear mathematical programming model can be used, we model the delay profile using a piecewise linear function [45] with pre-generated breaking points for each link $b_1, b_2, ..., b_n$. Then, for z_i $(0 \leq z_i \leq 1)$ as piece-wise linear slot weight, the rate x_l at link l can be written as:

$$x_l = z_i b_i + (1 - z_i) b_{i+1} \tag{3.2}$$

Since we have a linear function in the interval $b_i \leq x_l \leq b_{i+1}$, we can estimate the delay on the link in 3.3 as follows:

$$D_l(x_l) = z_i D_l(b_i) + (1 - z_i) D_l(b_{i+1}) \tag{3.3}$$

For each VNFs request r we calculate a set of shortest paths from the network gateway to the corresponding access router. Based on those pre-calculated shortest paths per request in the network, we define two binary variables where ω_{kp} and ζ_{pl} represent respectively the nodes and the links belonging to a path p,

$$\omega_{kp} = \begin{cases} 1 & \text{if node } k \in \text{path } p. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.4}$$

$$\zeta_{pl} = \begin{cases} 1 & \text{if link } l \in \text{path } p. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.5}$$

With u_{rp} we denote the rate that is allocated for request r at path p and d_r express the rate requirement for request r. We have two decision variables: v_{rk} representing the node k where all VNFs of a request r are hosted and ψ_{rp} the path p a request r is using,

$$v_{rk} = \begin{cases} 1 & \text{if VNFs of request } r \text{ hosted at node } k. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.6}$$

$$\psi_{rp} = \begin{cases} 1 & \text{if request } r \text{ use SP } p. \\ 0 & \text{Otherwise.} \end{cases} \tag{3.7}$$

3.3.2 Variables and Parameters

The main objective of this work is to minimize the overall delay in the network by selecting the optimal path. The choice for the routing path to be followed for a network service chain has to consider VNFs requirements as well as network available resources. Table 3.1 summarizes the variables and parameters used in this work which are similar to those defined in [57] research work.

Table 3.1 Description of variables for MILP formulation

Parameter	Domain	Description
x_l	$x_l \in [b_i, b_{i+1}]$	Total traffic in link l
D_l	$D_l \in \mathbb{R}_{>0}$	Delay in link l
S_p	$S_p \in \mathbb{R}_{>0}$	Cost of a path p
C_l	$C_l \in \mathbb{R}_{>0}$	Capacity of link l
z_i	$z_i \in [0, 1]$	Weight of piece-wise linear slot
R	$R \in \mathbb{N}$	Set of requests
r	$r \in \mathbb{N}$	Defines a request
L	$L \in \mathbb{N}$	Set of links
l	$l \in \mathbb{N}$	Defines a link
P	$P \in \mathbb{N}$	Set of paths
p	$p \in \mathbb{N}$	Defines a path
K	$K \in \mathbb{N}$	Set of nodes
k	$k \in \mathbb{N}$	Defines a node
η_r	$\eta_r \in \mathbb{R}_{>0}$	Resource requirement for a request r
Γ_k	$\Gamma_k \in \mathbb{R}_{>0}$	Available resource for node k
ω_{kp}	$\omega_{kp} \in \{0, 1\}$	Equal to 1 if node k belong to shortest path p
ζ_{pl}	$\zeta_{pl} \in \{0, 1\}$	Equal to 1 if link l belong to shortest path p
d_r	$d_r \in \mathbb{R}_{>0}$	Required rate for request r
u_{rp}	$u_{rp} \in [0, (C_l - x_l)]$	Allocated rate for request r in path p
v_{rk}	$v_{rk} \in \{0, 1\}$	Equal to 1 if VNFs of r are hosted in k
ψ_{rp}	$\psi_{rp} \in \{0, 1\}$	Equal to 1 if request r use path p
w_1	$w_1 \in [0, 1]$	Weight for delay optimization function
w_2	$w_2 \in [0, 1]$	Weight for routing cost optimization function

3.3.3 Mathematical Programming Formulation

Based on the delay definition in 3.3 and considering the path assigned to a service request and the delay of the links composing the chosen path (in other words the cost between two virtual machines), we define the objective function of minimizing the overall link delays in equation (3.8). In our case the cost of mapping the VNFs to the nodes is not considered since we are assuming that the functions are already placed

in the network.

$$\min_{\psi_{rp},v_{rk}} \sum_{l\in\mathbf{L}} \sum_{n\in\mathbf{N}} z_{n,l} D_l(b_n) \tag{3.8}$$

$$\text{s.t. } z_{1,l}b_1 + z_{2,l}b_2 + \cdots + z_{n,l}b_n = x_l \,\forall l \in \mathbf{L} \tag{3.9a}$$

$$z_{1,l} + z_{2,l} + \cdots + z_{n,l} = 1 \,\forall l \in \mathbf{L} \tag{3.9b}$$

$$\sum_{r\in\mathbf{R}} \sum_{p\in\mathbf{P}} u_{rp}\zeta_{pl}\psi_{rp} \leq C_l - x_l \,\forall l \in \mathbf{L} \tag{3.10}$$

$$\sum_{p\in\mathbf{P}} \psi_{rp} u_{rp} = d_r \,\forall r \in \mathbf{R} \tag{3.11}$$

$$\sum_{p\in\mathbf{P}} \psi_{rp} = 1 \,\forall r \in \mathbf{R} \tag{3.12}$$

$$\sum_{k\in\mathbf{K}} v_{rk} = 1 \,\forall r \in \mathbf{R} \tag{3.13}$$

$$\sum_{p\in\mathbf{P}} \sum_{k\in\mathbf{K}} \psi_{rp} v_{rk} w_{kp} = 1 \,\forall r \in \mathbf{R} \tag{3.14}$$

$$\sum_{r\in\mathbf{R}} \sum_{p\in\mathbf{P}} \eta_r v_{rk} \psi_{rp} w_{kp} \leq \Gamma_k \,\forall k \in \mathbf{K} \tag{3.15}$$

$$v_{rk}, \psi_{rp}, \zeta_{pl}, \omega_{kp} \in \{0, 1\} \tag{3.16}$$

$$z_{1,l}, z_{2,l}, \ldots, z_{n,l} u_{rp} \geq 0 \tag{3.17}$$

Constraints (3.9a) and (3.9b) ensure the piecewise linear approximation of the delay function using λ-formulation. (3.10) ensures that the capacity of the link is not exceeded. (3.11) ensures that the total rate allocated in all the paths is equal to the required rate for each request. (3.12) ensures that each request is mapped to one path. (3.13) ensures that all the VNFs of a request r are hosted in one node. Constraint (3.14) ensures that each request r using a path p, their VNFs are mapped in a node $k \in p$. And constraint (3.15) ensures that the capacity of a node k is not violated.

In order to linearize the optimization problem, in constraints (3.14) and (3.15), we replace the product of two binary decision variables [45] $v_{rk}\psi_{rp}$ with a new binary variable y_{rpk} where $y_{rpk} = v_{rk}\psi_{rp}$.
We define y_{rpk} as follows:

$$y_{rpk} = \begin{cases} 1 & \text{if request } r \text{ use SP } p \text{ and VNFs} \\ & \text{of request } r \text{ hosted at node } k \\ 0 & \text{Otherwise.} \end{cases} . \tag{3.18}$$

To linearize the constraints (3.14) and (3.15), we eliminate the non linear term $v_{rk}\psi_{rp}$ by replacing the two constraints in the defined optimization problem formulation with the following set of constraints:

$$\sum_{p\in\mathbf{P}}\sum_{k\in\mathbf{K}} y_{rpk}w_{kp} = 1 \,\forall r \in \mathbf{R} \tag{3.19}$$

$$\sum_{r\in\mathbf{R}}\sum_{p\in\mathbf{P}} \eta_r y_{rpk}w_{kp} \leq \Gamma_k \,\forall k \in \mathbf{K} \tag{3.20}$$

Note that constraints (3.21) (3.22) and (3.23) force the binary variable y_{rpk} to take the value of $\psi_{rp}v_{rk}$.

$$\sum_{p\in\mathbf{P}} y_{rpk} \leq v_{rk} \,\forall r \in \mathbf{R} \,\forall k \in \mathbf{K} \tag{3.21}$$

$$\sum_{k\in\mathbf{K}} y_{rpk} \leq \psi_{rp} \,\forall r \in \mathbf{R} \,\forall p \in \mathbf{P} \tag{3.22}$$

$$y_{rpk} \geq \psi_{rp} + v_{rk} - 1 \,\forall r \in \mathbf{R} \,\forall p \in \mathbf{P} \,\forall k \in \mathbf{K} \tag{3.23}$$

3.3.4 Multi Objective Optimization

In this part we introduce a new objective to ensure that the utilization of shortest paths are distributed equally and to avoid overloading link(s). A similar approach in term of joint objectives is presented in [59] by using a multi-objective approach to solve VNFs chain placement and routing problem with the joint objectives of minimizing the overall delay and VNFs chain placement cost. In addition to the first optimization problem we minimize the routing cost that we define based on the number of hops in the path p and the capacity of each link and we represent this cost by S_p. In order to process the objectives of minimizing delay and routing cost simultaneously, we

propose a multi-objective optimization (MOO) using a weighted sum approach [60]. We define the first weight for delay optimization as w_1 and the second weight for routing cost optimization as w_2 based on Pareto frontier optimality in order to find a trade-off between the delay and the routing cost. By doing so, we can transform equation (3.8) to the new multi objective function represented in equation (3.24).

$$\min_{\psi_{rp}, v_{rk}} w_1 \sum_{l \in \mathbf{L}} \sum_{n \in \mathbf{N}} z_{n,l} D_l(b_n) + w_2 \sum_{r \in \mathbf{R}} \sum_{p \in \mathbf{P}} \psi_{rp} S_p \tag{3.24}$$

3.4 Evaluation

We evaluate the effectiveness of our model and MILP formulation on a GEANT network topology [61] with has 22 nodes and 64 links. We assume that each link has a total capacity of a maximum of $2Gbps$.

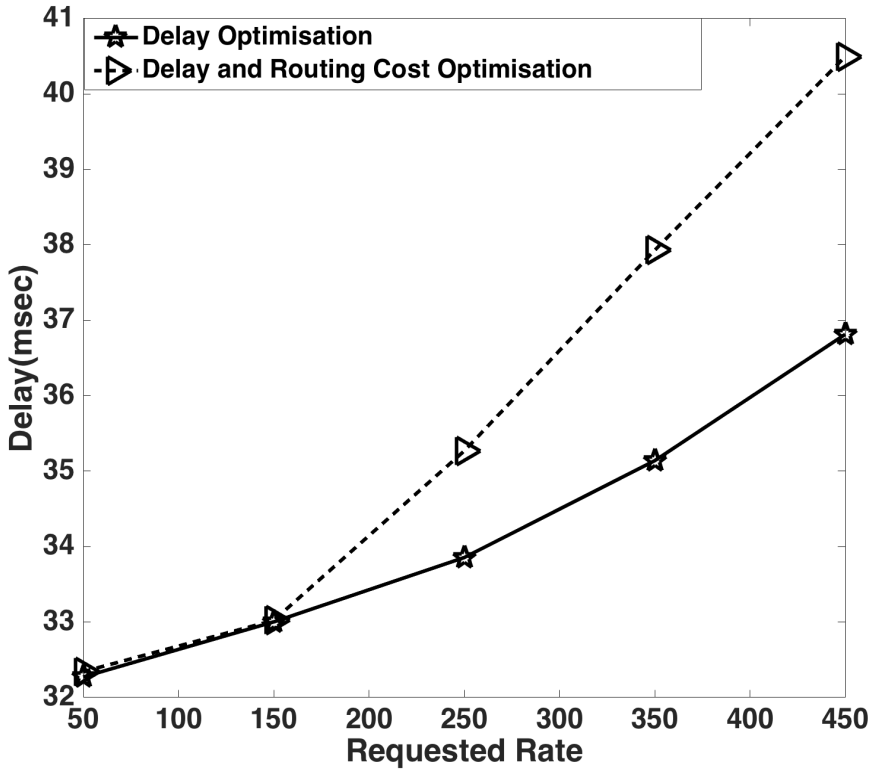

Figure 3.1: Impact of requested rate (Mbps) on delay (msec)

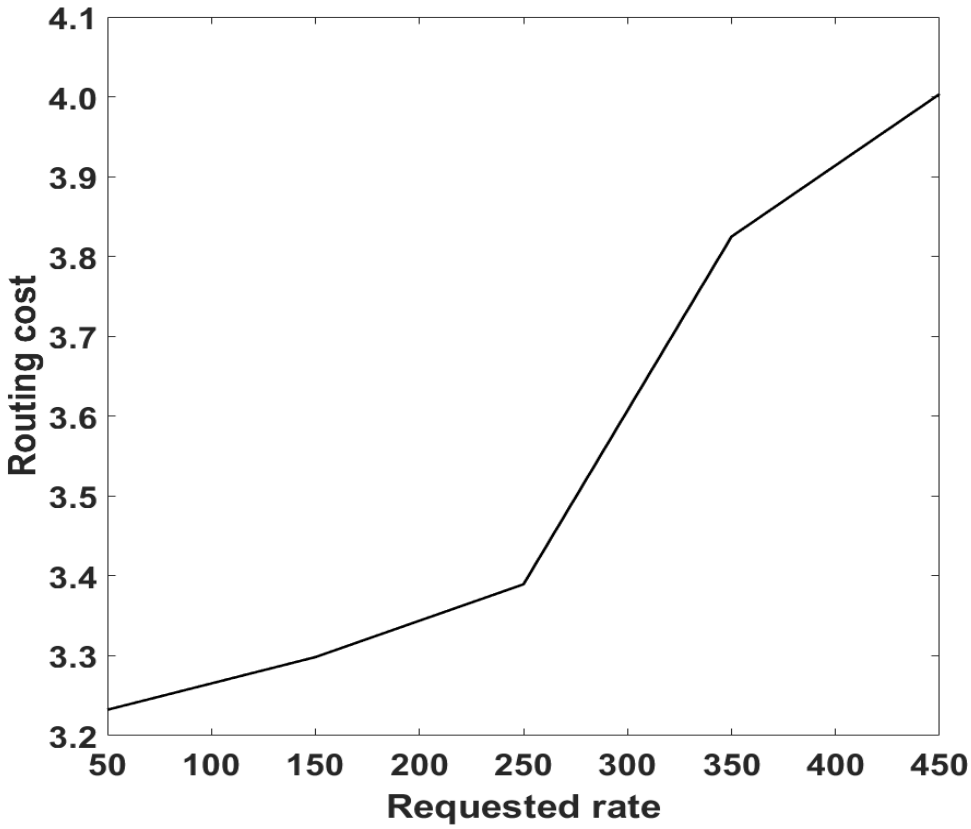

Figure 3.2: Impact of requested rate (Mbps) on routing cost

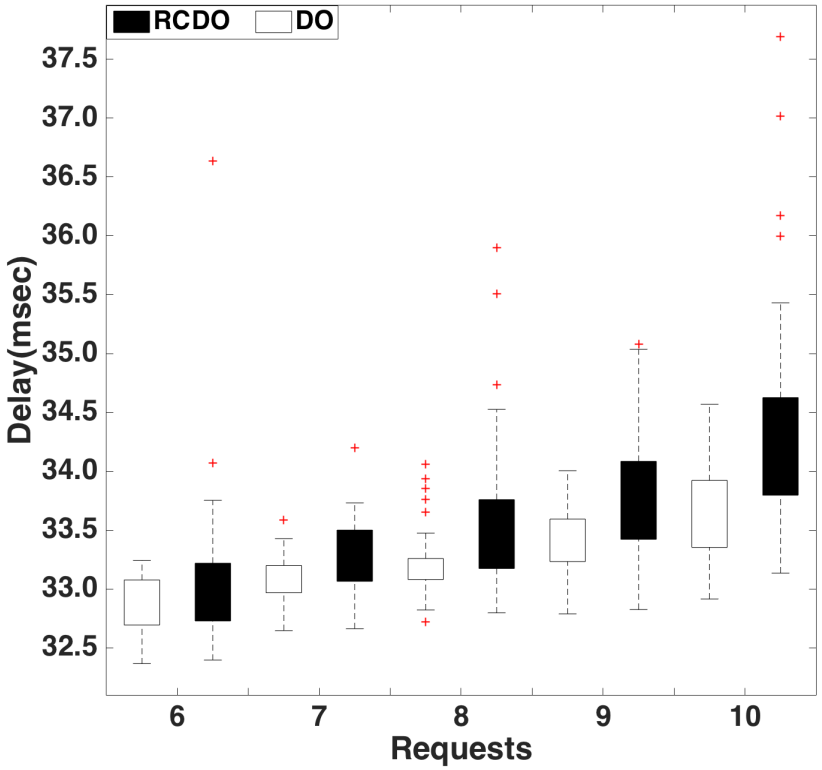

Figure 3.3: Impact of number of accepted requests on delay (msecs)

For the first evaluation, we limit ourselves to only changing the requested rate from $50Mbps$ (minimum rate) to $450Mbps$ (maximum rate). We assign a weight to each link based on their available capacities, where the link weight is defined as inversely proportional to its capacity as described in [62] within the range $(1/C_l, 1)$. For routing we use Dijikstra algorithm to find the two first shortest paths from an entry node to a sub-list of 8 destination nodes for each VNF-chain request. In case two or more requests have the same destination node the same shortest paths are defined for both demands to avoid congestion. On the other hand, in case of shortness of capacity in the first path a request will be assigned to the second shortest path. We also assume, without loss of generality, that all the service demands request a set of 5 VNFs (such as Network Address Translation, Deep Packet Inspection, Firewall). The cores needed are considered per service chain between 1 and 2 CPUs, knowing that we consider that all VNFs instances of a request are placed in one node and all the nodes have two available cores.

We propose two types of evaluations, the first one consists of observing the delay behaviour while increasing the requested rate for 10 requests for delay optimization

(DO) problem and joint routing cost and delay optimization (RCDO) problem. The results in Figure 3.1 show that the delay increases slightly. When the requested rate is between $50Mbps$ and $150Mbps$, choosing routes with low, medium or high cost does not affect the delay as we have enough available capacity. However, as the requested rate increases the number of common links increases between chosen paths to ensure a lower routing cost. Therefore, the delay value increases faster for delay and routing cost optimization solution compared to delay optimization solution.
We observe the routing cost in Figure 3.2 that allows us to examine the curve changing degree between delay and routing cost. The figure shows that the routing cost increases faster compared to the delay. The solution might choose a more expensive route in order to ensure a minimum delay. In other words, links with low costs are used until full. Therefore, in Figure 3.2 routing cost increases when the requested rate is more than $250Mbps$, then when it is more than $350Mbps$ because chosen paths have more common links and/or higher routing cost to insure a better delay and respects links capacity constraints. From these results, we conclude that the delay sharply increases when the requested rate is high due to the common links between defined shortest paths, in other words using a link by more than one request accelerates the inflation rate of delay.
The second evaluation consists of observing the network latency between the delay optimization problem and the joint delay and routing cost optimization problem. In some cases, when the number of requests is high, the solver is not able to find an optimal solution for the routing and the placement of the requests. Therefore, we consider the topology size and the available capacity of both links and cores and we run Monte Carlo simulation with 35 iterations for different number of requests from 6 to 10.

We consider two scenarios, in the first one we perform delay optimization (DO) only and in the second one we perform both routing cost and delay optimization (RCDO). As shown in Figure 3.3, the average delay increases, when the number of requests increases from 6 to 10 requests. Also, the delay difference between DO and RCDO is approximately 11% when 6 requests are accepted but note that the gap increases rapidly when the number of requests increases.

3.5 Conclusion

The inherent flexibility and potential capital/operational cost benefits of the NFV framework has so far encouraged widespread agreement within industrial circles in terms of its adoption. However the associated multi-scalar challenges of fully virtualized networks, especially those related to delay, reliability and robustness are yet to be fully understood. In this chapter the focus has been placed on delay minimization when considering chaining and routing of VNFs aiming to enrich emerging debates on issues related to ultra low latency communications. A joint routing and delay optimization framework has been presented by linearizing the inherent non-linear cost of the delay as a function of the utilization. It is true that the solution for the optimal chaining and routing with MILP limits the scale of the problem, since the calculation time will increase with the size of placement and routing of VNFs optimization problem, which could be a critical issue for larger coefficient matrices. The problem is combinatorial in nature entailing an NP-hard optimization problem. This issue was highlighted in different papers such as in [54] where a greedy algorithm have been proposed to overcome the scalability challenge caused by computation time. This issue can be solved by developing a heuristic solution to find a near optimal but scalable algorithm for VNFs placement and routing problem, as presented in the next chapter. The performance has been tested under various network conditions and the obtained results have shown promising performance.

Chapter 4

Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization

4.1 Introduction

In recent years, communication networks have been witnessing an exponential growth in user data traffic as well as an increase in the use of virtualization technologies. The deployment of network resources, the maintenance of hardware appliances and the never ending race for marketing new services have resulted for the network operators in an excessive OPEX, the ongoing costs a company pays to run its basic services, and CAPEX, the cost of expanding, upgrading and maintaining companies physical assets.

The ETSI describes a high-level NFV framework composed of three principal domains; VNFs, NFV infrastructure NFVI and NFV MANO. The group highlighted three key criteria to establish a high-level architecture framework: Decoupling as to complete the separation of hardware and software, flexibility in the automation and scalability of the network functions deployment, and dynamic operations in controlling the operational parameters of the network functions through control and monitoring the state of the network [63].
Cloud service providers such as Cisco, Google [64], Amazon [65] and Oracle own the virtual infrastructure and offer network services, infrastructure or applications from a shared infrastructure. Types of offered services can vary from Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) or host-based

applications. This allows users to get the required services and deploy their applications by paying only what they used, without any control or ownership. VNFs can be placed in different clouds and deployed in one or more Virtual Machine(s), different scenarios may require different scaling. For example, an application that needs to run faster in order to support more transactions per unit of time can be scaled vertically. On the other hand, horizontal scaling can be used for applications where the load can be spread across different VMs. In practise, a required service may be composed of more than one function, and traffic should go through the chain of VNFs in predefined order to provide the required service. Therefore, the placement and routing of VNFs is based on the required resources and affect both the quality of service and the overall cost for offering to the end user a specific service.

As the number of requests, the edge clouds and available VMs per edge cloud increases the problem of allocating resources becomes combinatorial in its nature. To this end, and in order to find optimal decision policies, we formulate the problem of a batch based network service chaining, routing, and placement. Based on the incoming virtual network requests and their requirements, such as delay tolerance, we consider a physical network infrastructure where different virtual networks have to be set up. We develop a mathematical programming formulation using MILP model to achieve an optimal solution, which consider vertical scaling for the purpose of minimizing network latency. For large networks, to accelerate the process of placement and routing, we use a scale-free heuristic algorithm in order to be able to provide a real-time allocation for a large number of requests. The proposed approach also takes into account the horizontal and vertical scaling of VMs. We also provide a performance comparison between the proposed heuristics and a simple greedy approach from the state-of-the-art [66].

The proposed model enables operators to increase the acceptance rate of strict delay requirement requests and to reduce rejected requests due to capacity constraints.
The main logic behind the proposed algorithm as shown in Figure 4.1 is a simple example of virtual network functions routing and placement in a small substrate network with 6 edge clouds is presented. We assume that we have two admitted requests, the related flows have different arrival rates. The figure shows different ways of scaling that we consider in this work. VNF_1 and VNF_2 are both required by request 1 and 2. We use horizontal scaling and vertical scaling for VNF_1 and VNF_2 instances, respectively. As illustrated in Table 4.1, we define a set of VNFs chain requests. The destination nodes are different, for request 1 and 2 destination nodes

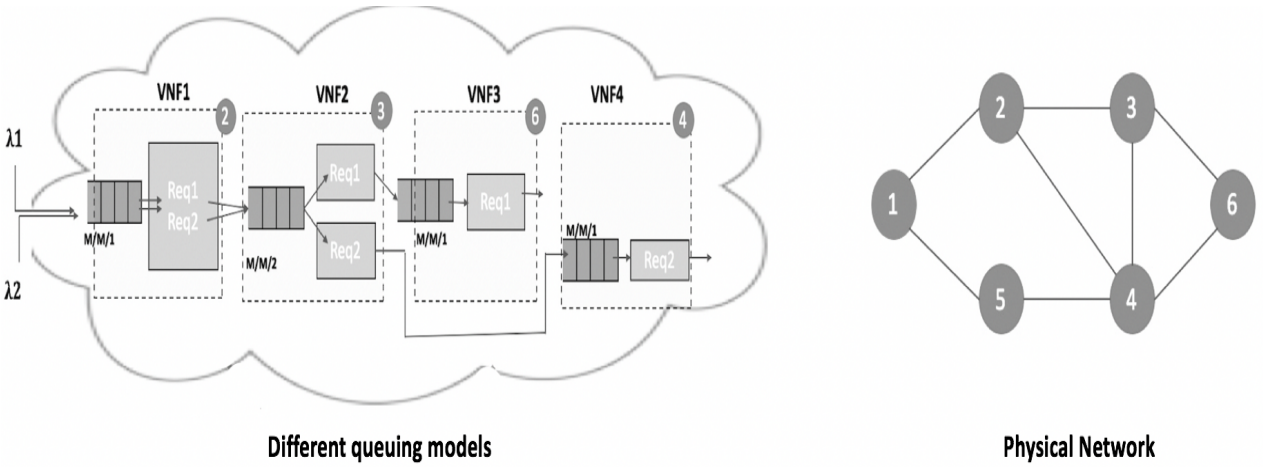

Figure 4.1: VNFs routing and placement example

are nodes 6 and 4, respectively. In all these cases, node 1 is considered as the gateway. Each request has a specific computing resource and bandwidth requirements, based on which we choose suitable physical links from two candidate paths selected in advance by k-shortest path algorithm. Then, we map VNFs following the algorithms presented in Section 4.5, by giving higher priority to VNFs requesting higher levels of computing resources and assign them to the physical node with the highest remain carrying capability. If two requests are sharing the same edge cloud to get the same service, we can either share the same instance with a high processing capacity between the request using vertical scaling as in the case of sharing VNF_1 instance between request 1 and request 2 in node 2. In the case of VNF_2, we use horizontal scaling by creating two instances of VNF_2 in node 3, one instance for request 1 and the second instance for request 2. We also present different queuing models in the same example, M/M/1 queuing model when one processing unit is serving the incoming requests and an M/M/m model to capture the delay when having two or more processing units. The rest of this chapter is organized as follows. In Section 4.2, we discuss the related research work and describe the proposed model in Section 4.3. A mathematical optimization model to minimize the overall latency is proposed in Section 4.4. We then propose a heuristic algorithm to solve the problem for larger instances scenario in Section 4.5. In Section 4.6, we discuss the experimental set-up and different results comparing between optimal and different suboptimal results for performance evaluation.

Table 4.1 Request requirements

Request Index	Candidate nodes	Requested functions	Carrying nodes	Candidate Paths
1	{2} {3} {4} {6}	1 2 3	{2} {3} {6}	$\{1 \rightarrow 2 \rightarrow 3 \rightarrow 6\}$ $\{1 \rightarrow 2 \rightarrow 4 \rightarrow 6\}$
2	{2} {3} {4}	1 2 4	{2} {3} {4}	$\{1 \rightarrow 2 \rightarrow 3 \rightarrow 4\}$ $\{1 \rightarrow 2 \rightarrow 4\}$

4.2 Background

Many previous research works have been focused in the area of VNF placement, chaining and routing by considering different metrics to increase network efficiency.
Different research works solved VM placement problem such as [67]. Abdelquoddouss Laghrissi et al. have defined mobile users behavior in terms of mobile service consumption. Also, the authors have introduced an Advanced Predictive Placement Algorithm (APPA) where the best locations are defined as the less utilized and the closest to most of the user equipments considering the overload of VMs, the data overload, and the QoS. Considering these metrics, a comparison with a number of VNF placement algorithms have been presented to measure the performance of the proposed APPA.

N. Kiji et al. have proposed an approach to VNF placement and routing model for multicast service chaining in [68]. The proposed merging multiple service paths based multicast service chaining model (MSC-M) consists of improving link usage and resource utilization. From one source node to multiple destination nodes, if two paths for different services carry the same data in one link, they can be merged into one path. MSC-M have resulted in decreasing the total cost by 28.7% compared to the traditional multicast service chaining model model. In [51], the authors have considered an autonomic resource management framework for virtual networks. They have argued that to ensure reliability, availability, and QoS requirements, advanced features of service offerings have to take place via an automation and elasticity of resource distribution and allocation. They have introduced an autonomic and distributed virtual network management resource management based on a reinforcement learning algorithm in order that the agents can learn progressively to enhance the performance of the resource management in virtual networks.

An interesting approach is proposed in [54] solving the problem of joint service placement and traffic steering incrementally. The authors have formulated the VNF placement and routing problem with the objective of minimizing both link and core resource utilization. For this purpose, they have modeled the problem using mathematical programming aiming at providing efficient placement of service chains while considering latency as a constraint rather than an objective to minimize.

Similarly, in [69], the authors have tackled the problem of VNF placement by considering two factors the paths between users and gateways in addition to feature mobility. This paper presents different VNF placement algorithms, such as Avoiding S-GW Relocation (A-SGWR) algorithm, which aim to minimize the Serving Gateway (S-GW) relocation overhead in a delay-constrained network. For the evaluation of this approach, authors have considered the delay of data packets delivery as one of the metrics. In [70], two efficient algorithms are presented to ensure the QoS and low cost deployment for vEPC/5G. One algorithm uses MILP to optimize the number of virtual resource instances of different VNFs of vEPC/5G core network and the second algorithm is based on coalitional game to place these instances over a federated cloud.

Chua et al. have proposed in [53] a SFC provisioning system referred to Stringer which enables virtual network providers to minimize the infrastructure resources and end-to-end delay. Three methods are used for the SFC provisioning system; a scalable round-robin heuristic, an optimization-based method and a queueing-theoretic model. This paper compares the performance of MIP with the heuristic method. The results show that the heuristic method outperforms the MIP significantly. However, this paper does not consider the routing cost.

A number of different approaches to the VNF placement problem consider delay as a requirement or observe the impact of different metrics on overall latency, with different optimization criteria such as reliability and load balancing. The authors in [71] have investigated the problem of virtual placement for optimal SFC deployment across distributed clouds. The authors have solved SFCs deployment focusing on VNFs placement through an affinity-based heuristic and minimize inter-cloud traffic and response time in a multi-cloud scenario as an ILP optimization problem. In this work, the latency is described as delay in the link and computational delays and modeled as M/D/1 and M/M/1 respectively.
The authors in [57] have presented an off-line approximation FAST-RACE algorithm for load balancing using multipath routing that decreases the latency and increases user demands. They have shown that using this method, the average delay of flows

decreases about 26% and increases user demands around 14% compared with those of the hop-count weight vector method for load balancing.
The authors in [56] have addressed the VNF scheduling problem and its respective resource optimization solutions. The authors have considered both VNF transmission and processing delays in this investigation. They have proposed a generic algorithm for solving the joint problem of VNF scheduling and virtual network resource allocation. They have evaluated the effectiveness of the proposed heuristic algorithm through a numerical method. They have shown that by dynamic allocation of bandwidth to virtual links, shorter scheduling can be achieved.
The work in [72] proposes a new resource allocation algorithm to enable energy-aware SFC in SDN-based virtual networks. The authors have mathematically formulated the problems of resource allocation of VNFs to traffic flows and flow routing as optimization problems with the aim of minimizing energy consumption and network reconfiguration overhead. They have proposed new heuristic algorithms for the above-mentioned optimization problems. They have shown that the proposed heuristic algorithms can offer sub-optimal solution near to the optimal solution as long as minimization of energy consumption is concerned.
In [73] the authors aim to find the optimal route in mobile wireless networks to minimize the total energy consumption. They model the problem as a joint optimization problem considering both the transmitting and receiving energy consumption. The efficiency of their framework was evaluated on a real-life network dataset and validated by three algorithms considering different delay constraints, which revealed lower energy consumption, optimizing transmitting and receiving cost and showing a trade-off between delay and the receiving energy in mobile wireless networks.
Bi, Zhu, Tian and Wang [74] have aimed to minimize the total number of VMs for a cluster-based three-tier virtualized applications by suggesting a flexible hybrid optimization. To do so, the authors have modeled the queue as a model of M/M/m system for the first tier and multiple M/M/1 for the remaining tiers. They have shown that under fine-grained resource provisioning, the optimum resource utilization can be achieved while maintaining average response time and request arrival time requirements.

In [66], the authors have considered inter-cloud latency and VNF response times to solve the problem of deploying SFCs as an ILP through an affinity-based heuristic. The latency is described as link delay and computational delay modeled as M/D/1 and M/M/1 respectively.
VMs are the most manageable entities sharing hardware resources [75] providing a

number of benefits such as isolation from hardware and other VMs [74]. Those VMs are scalable to meet the requirements of users/tenants in a virtualized environment. Scaling can vary according to the operator requirements such as traffic load, application type and the amount of input [76].
In [77], the authors have proposed an analytical model based on G/G/m queuing to estimate the mean response time of a VNF. The model can easily be extended to consider one or more service function chains. The validation of the model has been performed by computer simulation. The special case of the validation has been done for an LTE virtualized Mobility Management Entity (MME) with a three-tiered architecture. It has been shown that the proposed model has a computational complexity comparable to those used for analyzing Jackson networks and the estimation error of the mean response time is much lower than those of the considered baseline systems.
Rankothge et al. [78] have presented a resource allocation algorithm for VNFs based on Generic algorithms (GA). They have carried out an extensive analysis of two GA algorithms for both initial placement of VNFs and the scaling of existing VNFs for supporting traffic variation. It has been shown that the proposed GA algorithms outperform ILP resource allocation for a large number of VNFs in service function chains and the number of virtual machines where ILP takes several hours to process while GP takes only a few milliseconds.
As far as we are aware, none of the above papers have considered the joint optimization problem of VNFs routing and placement in a multi-clouds scenario. In this work, we formulate the optimization model considering multiple instances of the virtual functions across different edge clouds to serve flows of packets considering three VNFs models: single-feature single-request, single-feature multi-requests, and multi-features multi-requests [79].
We develop an optimization model to reduce the inter-cloud and link latency. The inter-cloud queuing delay and link delay are modeled as M/M/1. Later on, we present the problem as a Bin Packing problem solved by a standard heuristic approach following Best-fit Decreasing (BFD) method, where the inter-cloud queuing delay is modeled as M/M/1 and M/M/m and link delay is modeled as M/M/1. Additionally, we provide a performance comparison between heuristic and optimal solution and we compare the results of the proposed heuristic with those of random greedy.

4.3 Problem Description

As already stated above, the optimal VNFs chaining and routing problem is an area that has gained significant research attention and the problem itself falls within the NP-hard optimization problems. In this section, we set up the problem of minimizing the delay defined as inter-cloud and link queuing delay satisfying different constraints. We formulated the optimization model to route and assign VNFs to meet the service requests. All VNFs of a service request can be located at the same access/core location (the same edge cloud) or in different edge clouds. Furthermore, we assume that all the service requests are already admitted into the network, in other words, no admission control is considered in this work.

Data flows should visit different network functions depending on the required service, such as video optimizer, DIP, SBCs and Firewall in a specific order to be applied to the flow of data [52]. We define each service request as a chain of ordered functions similar to many research work such as in [58]. In our model, each request/service chain is associated with: a source node and a destination node in the network; a set of VNFs that needs to be executed on the flow and the arrival flow rate known in advance [80].

We assign a list of shortest path to each request using Dijkstra algorithm [41] in a weighted network following Yen's k-Shortest Path algorithm [81] explained in Appendix B. The weights assigned to links are positive, pre-defined by a service provider and can be related to link bandwidth, average link delay, or even the required power for transmission. Shortest paths are sorted by cost, the first shortest path is assigned as far as there is enough capacity to host the VNFs and enough bandwidth to transmit the flow through VNFs executed by VMs. In other words, we assume a set of pre-defined multiple shortest paths between end users and edge clouds as well as the network gateway. As a result, any multiple shortest path algorithm can be used in the proposed framework.

Every edge cloud is a pool of physical resources that can be shared through different VMs. Different VMs might offer distinct performance and execute the same service, this can be due to the heterogeneity of hardware. Since a VNF instance can adapt its capacity, VMs can be scaled up or scaled down.

We consider the fact that edge clouds have different geographical locations interconnected through virtual links; there is a traffic going between the nodes. For this, the capacity of links or edge clouds will be defined as the remaining capacity that can be used to solve VNFs routing and placement problem. We are going to compare three

models [79] in this chapter: single-feature single-request (SFSR) where a VNF instance can serve packets related to one request, single-feature multi-requests (SFMR) where a VNF instance, scaled up, can process more than one flow and multi-features multi-requests (MFMR) where different VNF instances, scaled horizontally, can process more than one flow all sharing the same buffer.
We will be using edge cloud and node interchangeably, where an edge cloud is defined by its remain capacity. The available resources of a node can be shared fully or partially, by one or different virtual machines. In the case where VMs are scaled horizontally, the set of one type of VNF instances will be homogeneous, but the set of VMs assigned to the same node can be heterogeneous if they are processing different VNFs. As illustrated in Figure 4.2, inter-cloud delay model can follow either M/M/1 (a) or M/M/m (b) queuing models where the queuing delay general equations are defined in (4.14) and (4.1) respectively.

In the first case (a), the edge cloud is considered as an aggregated processing unit, where the total processing capacity is the total of VM capacities assigned to it. However, in the second case (b), we consider each VM as the processing unit independently, which capacity is the resource portion assigned to each VM. In the case of low utilization, modeling the queuing delay as an M/M/1 model will result in under evaluating the delay compared to the M/M/m model representation. This clearly because one processing unit can process a request much faster than a small processing unit with only 20% capacity of the aggregated set of VMs in the model (a). This can be shown in Figure 4.3 where the delay in the case of M/M/1 queuing model is under evaluated compared to the delay we have modeled as M/M/m. However, the model results in a very close to a similar delay when the utilization is medium to high, this because one processing unit is expending its capacity between the requests as the flow of requests increases[82]. Therefore, the behaviour of one high capacity unit get closer to the performance of different small units with an equal total capacity. We consider in this work a medium to high arrival rate and utilization.

The delay in SFSR and SFMR can be modeled as M/M/1 queuing model where arrivals are determined by a Poisson process, in the same way, the delay in MFMR can be modeled as M/M/m for the reason that m VMs are processing the same function on different flows.

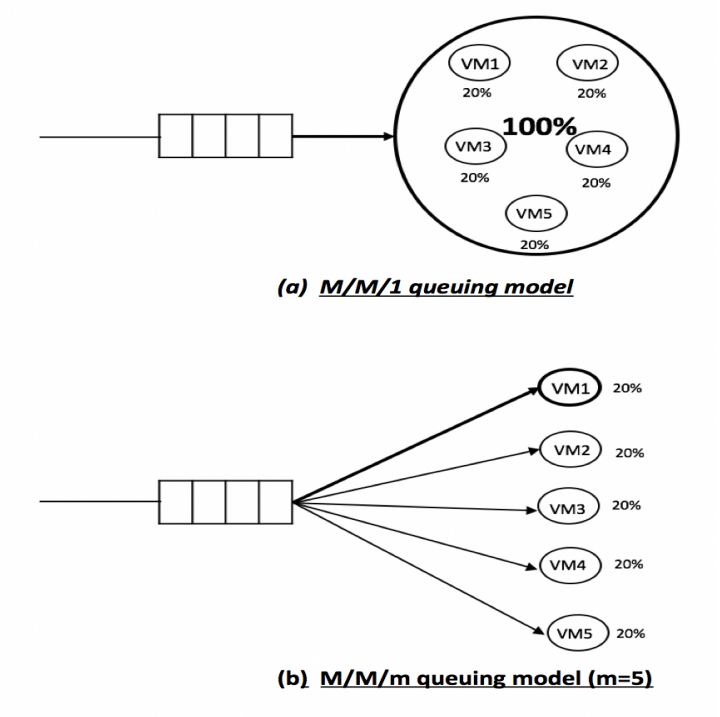

Figure 4.2: Horizontal scaling

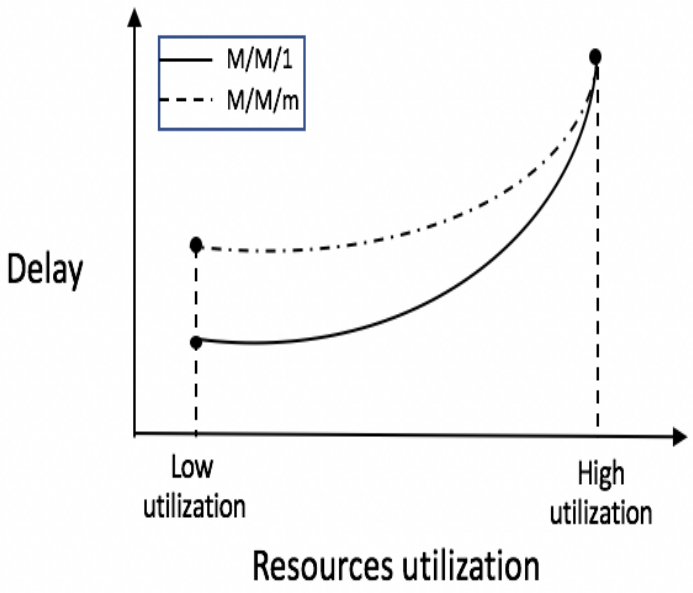

Figure 4.3: Delay modeling

4.3.1 VMs Scaling:

One way to scale virtual machines is vertical scaling (scaling up) that allow us to add more or less physical resources (CPU/Memory) to an existing virtual machine. This way of scaling allows us to resize the virtual machine by changing CPU or memory. Usually, vertical scaling requires downtime to add new resources and has defined limits by hardware.
On the other hand, horizontal scaling (scaling out) allow us to add more or less virtual entities to work as a single logical unit to adapt to network load changes. Based on resource demands we can dynamically add or reduce the number of VMs.

4.3.2 Queuing Theory

Traditionally, queuing theory is used to model servers and internet routers, to measure different metrics and improve network performance [83][84]. In this work we provide an amalgamation of queueing theory with integer programming in order to optimize the overall delay. To this end, we utilize two queuing models methods; the M/M/1 which is used to model link queues, whereas servers queues in the edge clouds are modelled using the M/M/m model. The difference between the two models is that with the M/M/m we assume that there are m available resources to run VNFs (i.e., m VMs) in the system that are independent. Similarly to the M/M/1 model, arrivals and servers service times follow exponential distribution with λ and μ parameters respectively. As defined in the queuing theory [85], the delay is formulated based on the definitions of the average processing time (4.2), the arrival rate (4.3), its ratio (4.4), the probability of a customer waiting in the queue (4.5) or none (4.6), and the average waiting time (4.7). The variables are summarized in Table 4.2 and the delay is formulated in equation (4.1) as:

$$D_{e,f} = \frac{1}{\mu_{ef}} + W_{ef} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.1}$$

To define the average waiting time W_{ef} of a packet in the queue (4.7) we define the average processing time and the arrival rate in equations (4.2) and (4.3) respectively.

$$\mu_{ef} = \frac{\sum_{i=1}^{m} \mu_{ief}}{\sum_{i=1}^{r} x_{ief}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.2}$$

Table 4.2 Description of queuing system notations

Parameter	Description
μ_e	Service rate (inverse of average service time)
λr	Arrival rate (inverse of average inter arrival time)
W_{ef}	Average customer waiting time in queue
ρ_{ef}	Ratio of arrival rate
P_{ef}	Probability that an arriving customer has to wait in queue
p^0_{ef}	Probability of no customers in the system

$$\lambda ef = \sum_{i=1}^{r} \lambda_i x_{ief} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.3}$$

$$\rho_{ef} = \frac{\lambda ef}{m\mu_{ef}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.4}$$

$$P_{ef} = \frac{p^0_{ef}(m\rho_{ef})^m}{m!(1-\rho_{ef})} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.5}$$

$$p^0_{ef} = \frac{1}{\sum_{k=0}^{m-1} \frac{(m\rho_{ef})^k}{k!} + \frac{(m\rho_{ef})^m}{m!(1-\rho_{ef})}} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.6}$$

$$W_{ef} = \frac{\rho_{ef} P_{ef}}{\lambda_{ef}(1-\rho_{ef})} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.7}$$

4.3.3 Virtual Network Functions Affinity:

Affinity and anti-affinity rules in NFV must be considered and added carefully in order to reduce communication costs between VNFs instances, ensure high availability, resilience, privacy and service performance [86]. In this context, two main aspects should be considered: modeling and describing the affinity rules and adapting the placement algorithm to respect the constraints [87].

Depending on the use case, there might be instances where we need to place a pair of VNFs on the same edge-cloud (e.g., VNFs exchanging a big amount of data). In this case, we should define affinity constraints to place the two or more VNFs in the same host [88]. In other cases, anti-affinity rules are considered to allow critical VNFs to run on different nodes (e.g., in the case of failure, it will be better to have different instances of the same function placed on different edge clouds or different

physical servers in the same edge cloud). Anti-affinity rules ensure the minimum cross interaction between VNFs running on the same server.

Based on the above discussion, We pre-initialize an affinity matrix that defines if two VNFs have a high affinity or a non-affinity relation. V_{ij} will be defining the affinity between VNF_i and VNF_j as follows:

$$V_{ij} = \begin{cases} 1 & \text{if } VNF_i \text{ and } VNF_j \text{ have a high affinity.} \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

To ensure that affinity between VNFs does not affect the performance of the network, we define the affinity constraint (4.9).

$$\sum_{i \in \mathbf{F}} \sum_{j \in \mathbf{F}} \sum_{r \in \mathbf{R}} V_{ij} x_{rei} x_{rej} = 1 \quad \forall e \in \mathbf{E} \tag{4.9}$$

4.4 System Model

In this section, we present our model as a mathematical programming formulation where key notations are described in Table 4.2 and Table 4.3. We set up the problem of minimizing inter-cloud and link delays in a multi-cloud scenario as a MILP optimization problem. The goals of our approach are to place VNFs, route workflows and assign requests to these flows to meet the service demands. We present our objectives combined with constraints, then we elaborate on how we linearized the objective function and nonlinear constraints.

4.4.1 Optimization Function

We will be modeling edge-cloud and link delay in the following subsections using the M/M/1 queuing model to optimize the delay. Packets for each flow request are enqueued in every edge cloud waiting to get processed by available VMs, and then in another queue to get transmitted through network links. Each Edge cloud can have none, one or different queues, depending on the number of VNF instances assigned to it.

We consider the M/M/1 queuing model for both link traffic and inter-cloud traffic where one VM is available to serve incoming traffic requiring a specific function.

We define the decision variable x to define the allocation of VNFs instances composing every service request, such that x_{ref}=1 or x_{ref}=0. The value 1 will be assigned if VNF

f of a request r is assigned to an edge cloud e consuming a portion of its resources; memory, network, and computing available resources. Decision variable ψ represents the assignment of one path p to one or different requests, $\psi_{rp} = 1$ means that request r will use path p and go through all nodes and links belonging to path p to get the requested services. Requests might share the same path or have common links. Both variables v_{ep} and ζ_{pl} define the nodes and the links belonging to path p respectively.

$$x_{ref} = \begin{cases} 1 & \text{if flow related to request } r \text{ go} \\ & \text{through edge cloud } e \text{ for VNF } f. \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

$$\psi_{rp} = \begin{cases} 1 & \text{if request } r \text{ use SP } p. \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

$$v_{ep} = \begin{cases} 1 & \text{if edge cloud } e \in \text{path } p. \\ 0 & \text{otherwise.} \end{cases} \tag{4.12}$$

$$\zeta_{pl} = \begin{cases} 1 & \text{if link } l \in \text{path } p. \\ 0 & \text{otherwise.} \end{cases} \tag{4.13}$$

Using Little Theorem [85], we define the delay in the inter-cloud in Equation 4.14 where h_{rf} is the processing capacity and λ_r is the arrival rate.

$$N_{ref} = \frac{1}{h_{rf} - \lambda_r} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.14}$$

Similarly, we define the link delay as follows, where C_l is the capacity of the link and T_l is the total traffic in link l.

$$L_l = \frac{1}{C_l - T_l} \quad \forall l \in \mathbf{L} \tag{4.15}$$

In order to minimize the overall delay from the gateway to the end-users, we solve the MILP formulation modeling both inter-cloud and link queuing delays. We use MILP Matlab tool and formulate the objective function in the equation (4.16) and applicable constraints in equations (4.17-4.26) based on the above definitions, the mathematical problem can be formulated as follows:

$$\min_{\psi_{rp}, x_{ref}} \sum_{l \in \mathbf{L}} \sum_{n \in \mathbf{N}} z_{ln} L_l(b_n) + \sum_{r \in \mathbf{R}} \sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} x_{ref} N_{ref} \tag{4.16}$$

$$\text{s.t. } z_{1,l}b_1 + z_{2,l}b_2 + \cdots + z_{n,l}b_n = x_{1,l} \quad \forall l \in \mathbf{L} \tag{4.17a}$$

$$z_{1,l} + z_{2,l} + \cdots + z_{n,l} = 1 \quad \forall l \in \mathbf{L} \tag{4.17b}$$

$$\sum_{r \in \mathbf{R}} \sum_{f \in \mathbf{F}} x_{ref} h_{r,f} \leq \mu_e \quad \forall e \in \mathbf{E} \tag{4.18}$$

$$x_{ref} \lambda_r \leq \omega_{ref} \mu_e \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.19}$$

$$x_{ref} h_{r,f} = \omega_{ref} \mu_e \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.20}$$

$$\sum_{r \in \mathbf{R}} \sum_{p \in \mathbf{P}} \lambda_r \zeta_{pl} \psi_{rp} \leq C_l \quad \forall l \in \mathbf{L} \tag{4.21}$$

$$\sum_{p \in \mathbf{P}} \psi_{rp} = 1 \quad \forall r \in \mathbf{R} \tag{4.22}$$

$$\sum_{e \in \mathbf{E}} \sum_{f \in \mathbf{F}} \sum_{p \in \mathbf{P}} x_{ref} \psi_{rp} v_{ep} \eta_{rf} = d_r \quad \forall r \in \mathbf{R} \tag{4.23}$$

$$\sum_{f \in \mathbf{F}} \sum_{r \in \mathbf{R}} \omega_{ref} A_f \leq R_e \quad \forall e \in \mathbf{E} \tag{4.24}$$

$$\sum_{f \in \mathbf{F}} \sum_{e \in \mathbf{E}} \sum_{l \in \mathbf{L}} \sum_{p \in \mathbf{P}} \frac{x_{ref}}{h_{rf} - \lambda_r} + \frac{\zeta_{pl} \psi_{rp}}{C_l - T_l} \leq T_r \quad \forall r \in \mathbf{E} \tag{4.25}$$

$$v_{ep}, \psi_{rp}, x_{ref}, \eta_{rf}, \zeta_{pl} \in \{0, 1\} \quad h_{rf}, \lambda_r, \mu_e \geq 0 \tag{4.26}$$

4.4.2 Explanation of the Optimization Problem Constraints

Constraints (4.17) ensure the piecewise linear approximation [45] for the link delay function using λ-formulation. Constraints (4.18), (4.19) and (4.20) ensure that the available capacity of the edge cloud e is not exceeded. In a similar manner, constraint (4.21) ensures that the link capacity is not exceeded. Constraint (4.22) enforces that each request r to be assigned to only one routing path p. Constraint (4.23) ensures that when a request r is using a path p, all VNFs required for this request are mapped into edge nodes belonging to that chosen path p. Constraint (4.24) makes

Table 4.3 Description of variables

Parameter	Domain	Description
$h_{r,f}$	$h_{r,f} \in \mathbb{R}_{>0}$	Processing rate of function f for request r
$N_{r,e,f}$	$N_{r,e,f} \in \mathbb{R}_{>0}$	Delay in edge cloud e related to function f
L_l	$L_l \in \mathbb{R}_{>0}$	Delay in link l
T_l	$T_l \in \mathbb{R}_{>0}$	Total traffic in link l
C_l	$C_l \in \mathbb{R}_{>0}$	Remain Capacity of link l
S_p	$S_p \in \mathbb{R}_{>0}$	Cost of a path p
R	$R \in \mathbb{N}$	Set of requests
r	$r \in \mathbb{N}$	Defines a request
L	$L \in \mathbb{N}$	Set of links
l	$l \in \mathbb{N}$	Defines a link
P	$P \in \mathbb{N}$	Set of paths
p	$p \in \mathbb{N}$	Defines a path
E	$E \in \mathbb{N}$	Set of Edge clouds
e	$e \in \mathbb{N}$	Defines an edge cloud
F	$F \in \mathbb{N}$	Set of functions
f	$f \in \mathbb{N}$	Defines a function
N	$N \in \mathbb{N}$	Set of breaking points
n	$n \in \mathbb{N}$	Defines a breaking point
η_{rf}	$\eta_{rf} \in \{0,1\}$	Equal to 1 if request r requests function f
x_{ref}	$x_{ref} \in \{0,1\}$	Equal to 1 if request r goes through edge cloud e for function f
ψ_{rp}	$\psi_{rp} \in \{0,1\}$	Equal to 1 request r is assigned to shortest path p
v_{ep}	$v_{ep} \in \{0,1\}$	Equal to 1 if edge cloud e belong to shortest path p
ω_{ref}	$\omega_{ref} \in [0,1]$	Integer variable defining the utilization of an edge cloud e capacity to host a VNF instance f for a request r
d_r	$d_r \in \mathbb{R}_{>0}$	Number of functions required by request r
μ_e	$\mu_e \in \mathbb{R}_{>0}$	Average processing rate of an edge cloud e
λ_r	$\lambda_r \in \mathbb{R}_{>0}$	Arrival rate of request r flow
ζ_{pl}	$\zeta_{pl} \in \{0,1\}$	Equal to 1 if link l belong to shortest path p
A_f	$A_f \in \mathbb{R}_{>0}$	Resource demand of each service instance of VNF f
R_e	$R_e \in \mathbb{R}_{>0}$	Resource capacity of a computing node e
T_r	$T_r \in \mathbb{R}_{>0}$	Delay tolerance of a request r

sure that a VNF f is placed at an edge cloud e with sufficient resource capacity. Finally, constraint (4.25) considers each request requirement in terms of delay tolerance.

4.4.3 Linearization of the Proposed MILP

In order to linearize the optimization problem, in constraint (4.23), we replace the product of two binary decision variables $x_{ref}\psi_{rp}$ with a binary variable u_{refp} where u_{refp}=$x_{ref}\psi_{rp}$.

To linearize the constraint (4.23), we eliminate the non-linear term $x_{ref}\psi_{rp}$ by replacing the product as follow:

$$\sum_{e\in\mathbf{E}}\sum_{f\in\mathbf{F}}\sum_{p\in\mathbf{P}} u_{refp}v_{ep}\eta_{rf} = d_r \quad \forall r \in \mathbf{R} \tag{4.27}$$

Note that constraints (4.28) (4.29) and (4.30) force the binary variable u_{refp} to take the value of $x_{ref}\psi_{rp}$.

$$\sum_{p\in\mathbf{P}} u_{refp} \leq x_{ref} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.28}$$

$$\sum_{p\in\mathbf{P}} u_{refp} \leq \psi_{rp} \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \tag{4.29}$$

$$u_{refp} \geq x_{ref} + \psi_{rp} - 1 \quad \forall r \in \mathbf{R} \quad \forall e \in \mathbf{E} \quad \forall f \in \mathbf{F} \quad \forall p \in \mathbf{P} \tag{4.30}$$

4.5 Heuristic Based Algorithms

In this part, we present different heuristic approaches that address the same goals as the optimal one. We are using heuristic approaches to generate competitive sub-optimal solutions that are able to scale with the size of the problem and reduce the computational complexity. In the first place, we compare the proposed heuristic with a random greedy method, and secondly with a greedy approach based on FFD (first-fit decreasing method). Algorithm 5 shows the steps for the greedy, before we iterate through all required VNFs instances, we group the VNFs needed to satisfy all requests

in one list. Respecting capacity constraints, we allocate VNFs instances at the appropriate node, placing a maximum number of instances in the chosen node, before we move to the next node. The second step of the algorithm consists of assigning every request to a node or multiple nodes hosting the required VNFs and satisfying the capacity constraints. We finally define the routing path following the shortest path approach, to link between the gateway, the chosen nodes, and the destination node.

Algorithm 1 Input Parameter

Require: A connected edge clouds topology G(N x N) and list of requests R(R x V)
AllVnfsPlaced = false
Read λ_r, μ_e and C_l.
Read RC(R x 1) as requests required capacity
Sort R Desc
//N= Number of Edge clouds
Read LRC(N x N) as Links remain capacity
Read NRC(N x 1) as Nodes remain capacity
Sort NRC Desc
//V= Number of VNFs instances
Read VRR(V x R) as VNFs instances required capacity
//P= Number of shortest paths
Construct SP(R x P) as shortest paths list.
Construct SPmin(R x P) as minimum link bandwidth in the path

4.5.1 Initialization Algorithm

Algorithm 1 describes the steps to initialize all input parameters and completing the routing processing step. We first generate a random connected graph where each edge cloud and every link is defined by its remaining capacity. We assume a number of accepted service requests, each defined by an arrival rate a source and a destination. VNFs offer different services and process differently, for this we set up the required amount of resources to instantiate every VNF. Since the routing algorithm is the same for horizontal and vertical scaling, we set up a matrix of shortest paths. For each service demand, we calculate the 3 first shortest paths [89] with enough capacity to handle the related traffic flow. This pre-processing is common to heuristic algorithms presented in the following sub-sections.

Algorithm 2 HSFSR & HMFMR (Horizontal scaling)

```
Foreach (Request r in RC)
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
    Foreach (Node n in NRC)
    if (n ⊂ SP(r,p) and VRR(v,r) ⩽ NRCcopy(n) and RC(r) ⩽ SPmin(r,p))
      Assign VNF instance v to Node n
      Update NRCcopy and LRCcopy
      Break;
    EndFor
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Sort NRC Desc
  Copy LRCcopy in LRC
EndFor
```

Algorithm 3 HSFMR (Vertical scaling)

```
Foreach (Request r in RC)
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
      Foreach (Node n in NRC)
      if (n ⊂ SP(r,p) & VRR(v,r) ⩽ NRCcopy(n) & RC(r) ⩽ SPmin(r,p))
        Assign VNF instance v to Node n
        If (VNF f have been assigned to Node n )
          Assign more resources to VM
        else
          New VM will process VNF f
        endIf
        Update NRCcopy and LRCcopy
        Break;
      EndFor
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Sort NRC Desc
  Copy LRCcopy in LRC
EndFor
```

For the next step, we are using the initialization algorithm to generate the input parameters. We will be defining three algorithms: a heuristic algorithm for horizontal scaling, a heuristic algorithm for vertical scaling and a Random fit greedy algorithm where the random assignment will help us to measure the impact of our approaches on latency.

4.5.2 Horizontal Scaling Algorithm

Algorithm 2 describes the steps for the heuristic method used for two different cases. In the first case of SFSR, each VNF instance may serve only one request at a time. We follow an M/M/1 queuing model, where we have one queue per VNF instance, the flow(s) of data processed by the VNF instance are related to one request. In the second case of MFMR, VNFs instances can serve different tenants, different service requests sharing the same VNF. We follow an M/M/m queuing model [85] and we group VMs per function type, e.g. packets related to flows requiring the same service and assigned to the same node will share the same buffer. Furthermore, the number of VNFs instances will be the same as the number of different service requests. The heuristic method iterates through all requests, for each we iterate through all requested VNFs in order to place the one with the highest resource demands to the edge cloud with the highest remaining capacity following the BFD approach. At the end of every iteration, if not all the VNFs of a specific request are placed we start over using the next available path. The main difference between heuristic based SFSR (HSFSR) and heuristic MFMR (HMFMR) algorithms is in the cost measurement (calculation of the inter-cloud delay).

4.5.3 Vertical Scaling Algorithm

In Algorithm 3, each VNF instance can serve different requests at the same time. One buffer will be hosting different flows queuing to have a similar processing by the same VNF instance. Similarly to the first heuristic, we are following the same approach of BFD. After assigning the VNFs to edge clouds, VMs are scaled vertically in order to serve different flows related to different requests. In the case two or more requests have a function in common and have been assigned to the same edge cloud, they will share the same VM (the same VNF instance). Instead of assigning a VNF to another VM in the same edge cloud, we will be assigning more resources to the same VM to

be shared. To cope with a higher number of demands without creating additional VMs. This type of scaling can be used to avoid VMs under-utilization.

Algorithm 4 Randomized heuristic algorithm

```
Foreach (Request r in RC)
    Choose a path from the SR list
  While (AllVnfsPlaced = false)
{
    Copy NRC in NRCcopy
    Copy LRC in LRCcopy
    Foreach (VNFs v in VRR(v,r))
    Scan NRCcopy for a node to accommodate the VNF instance/VM
    if (such node is found)
      Assign VNF instance v to Node n
      Update NRCcopy and LRCcopy
      Break;
    EndFor
    if (All VNFs of R are placed)
      AllVnfsPlaced = true
    EndIf
    Go to next Shortest path;
}
  Copy NRCcopy in NRC
  Copy LRCcopy in LRC
EndFor
```

4.5.4 Random Placement and Routing Algorithm

Additionally, with the random routing and placement algorithm, for each service demand, the algorithm selects one path randomly to be assigned to a request. Then, it randomly choose one of the nodes with sufficient capacity for the placement of VNFs. The results will be restrained by the constraints defined in the LP approach and compared with the heuristic algorithm outputs.

4.6 Experimental Setup and Results

In this section, we observe and analyze the behavior of the proposed heuristic based on three different models: SFSR, SFMR, and MFMR. We compare suboptimal approaches and present HSFSR, heuristic-based SFMR (HSFMR) and HMFMR results,

Algorithm 5 Greedy heuristic

```
Foreach (VF instance v in V)
Foreach (edge cloud e in NRC)
  While (AllVnfsPlaced = false and NRC(e) ≥ VRR(v) )
  {
    if (constraints are satisfied and node has enough
      capacity)
      Assign VNF instance v to Node e
    EndIf
  }
EndFor
EndFor

Foreach (Request r in RC)
Foreach (edge cloud e in NRC)
Foreach (VF instance v in V)
    if (v instance is installed in edge cloud e)
      Assign request r to Node e
    EndIf
EndFor
EndFor
EndFor
```

we also compare their results with those of the MILP based solution. Furthermore, we show that the proposed heuristics allow us to increase the size of the problem solved compared to the MILP based solution. Thus, they allow a scale-free operation, amenable to run in large network topologies with an increased number of requests.

Table 4.4 Virtual processing times of virtual network functions used in our evaluation

Network Function	Processing time
Load Balancer	0.647.5 pps
Firewall	7.0771 pps
VPN Function	1.6385 pps

For instance, we have evaluated our approach on a random 28 nodes topology network, each with total CPU capacity of 100%. To build our topology, we consider one gateway and several destination edge clouds in a random connected graph, having n possible vertices and N edges, chosen randomly with equal edges probabilities [90][91]. To build the graph we follow the theory of random Walk [92], where we select a starting point node, we select a neighbor for it at random and move to this neighbor; then we select a neighbor node for this point at random, and move to it.

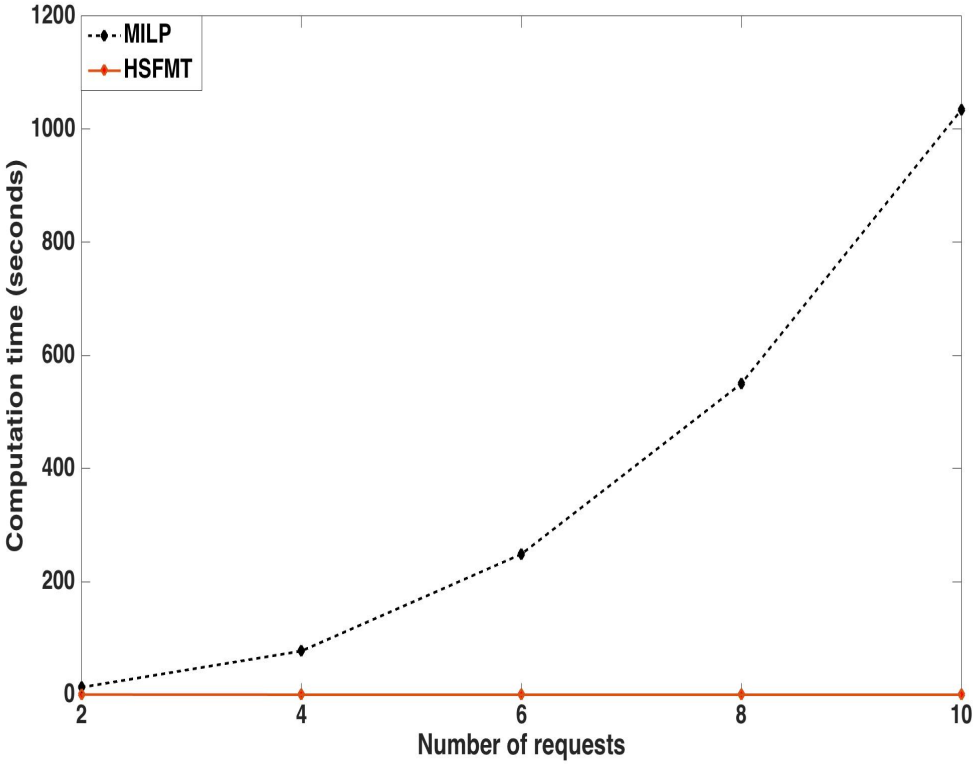

Figure 4.4: Computational time (s)

Each request is defined by a source gateway to a specific destination node in the graph.

For simplicity, we assume that a packet size is 500 bytes, a request arrival rate is assumed to be 1 to 100 packets per second (pps) [42] and VNFs instances have different processing service rates [91] as illustrated in Table 4.4. We vary link transmission capacities randomly from {2, 20, 200, 510} Kpps to 2 Mpps [71].

The proposed MILP framework takes 144 seconds to find optimal solutions for four service chains composed of three services chosen randomly from a list of five different VNFs (Load Balancer, Firewall, Intrusion Detection System (IDS), DPI, virtual private network (VPN) function) where 4 GB RAM used by the optimization solver, while the heuristic methods take only three seconds to run for 220 requests.

To compare the optimal solutions coming from the MILP framework with the heuristic results, we focus on small scale scenarios. This is because, as expected, integer programming suffers from the curse of dimensionality, hence the calculation time increases exponentially with a linear increase of the size of the problem [54]. To in-

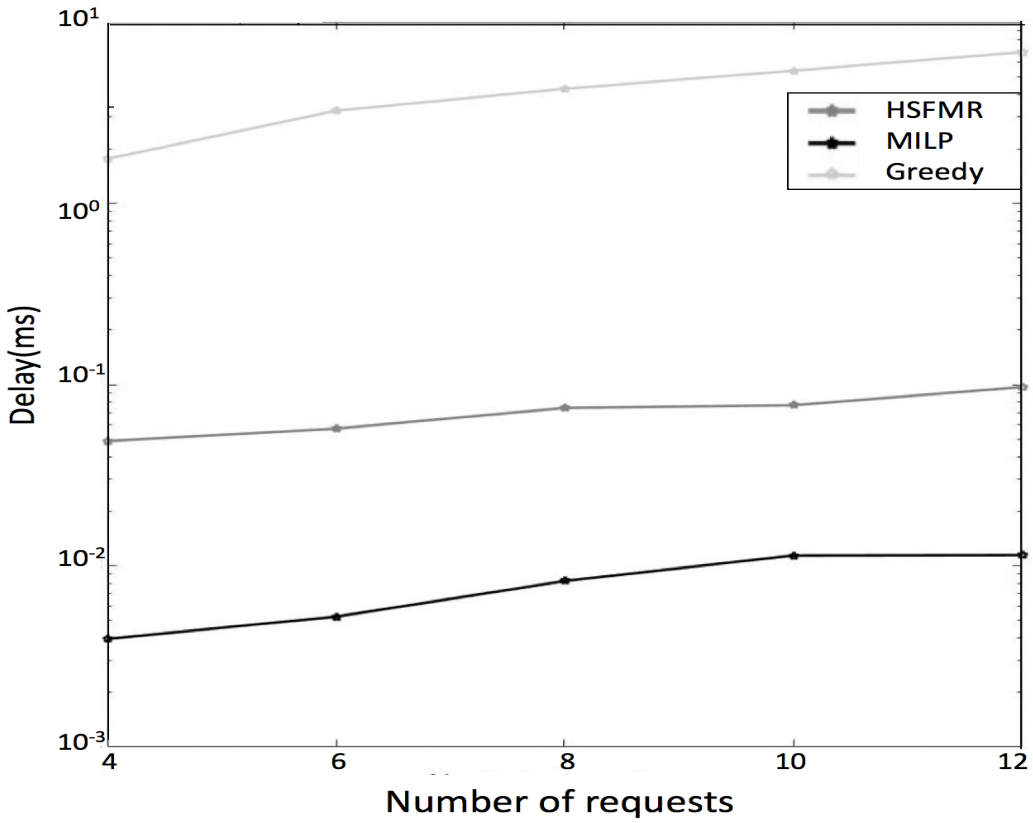

Figure 4.5: Heuristic vs. optimal (varying number of requests)

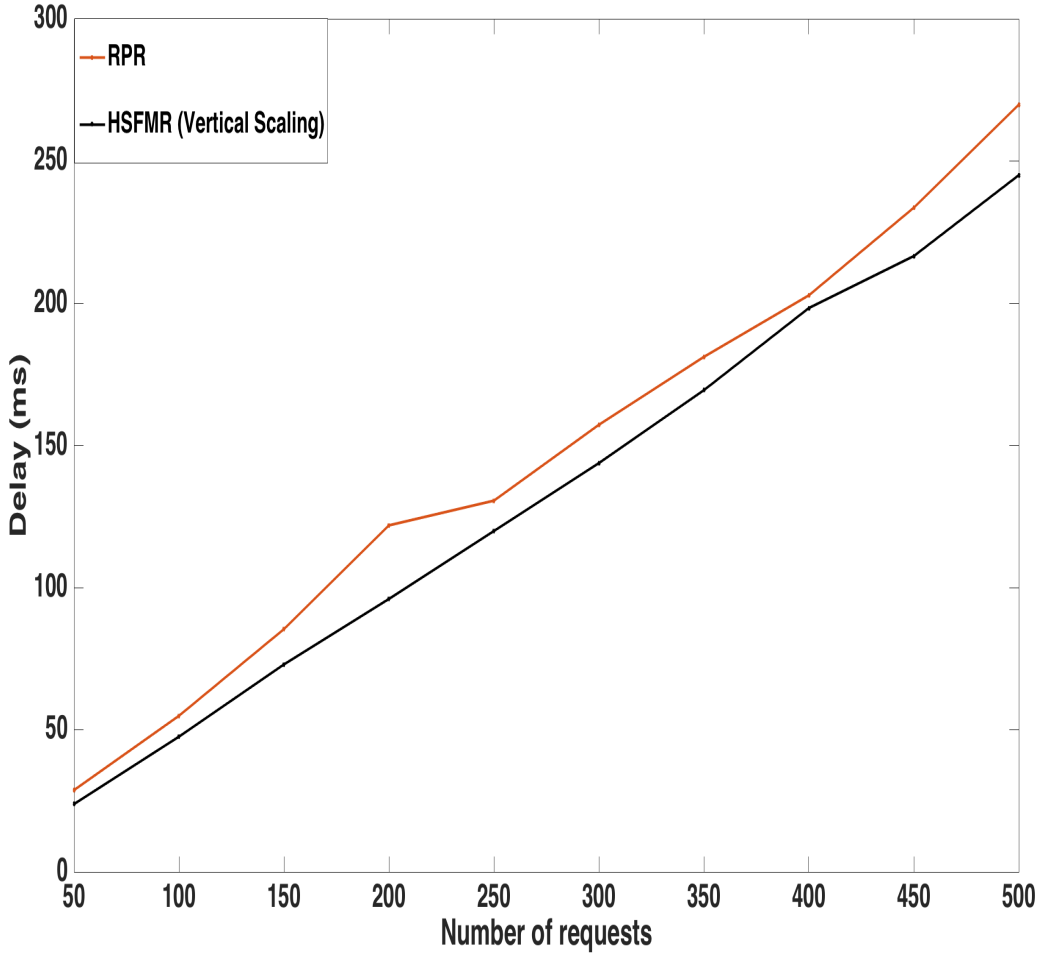

Figure 4.6: Heuristic vs. Random greedy

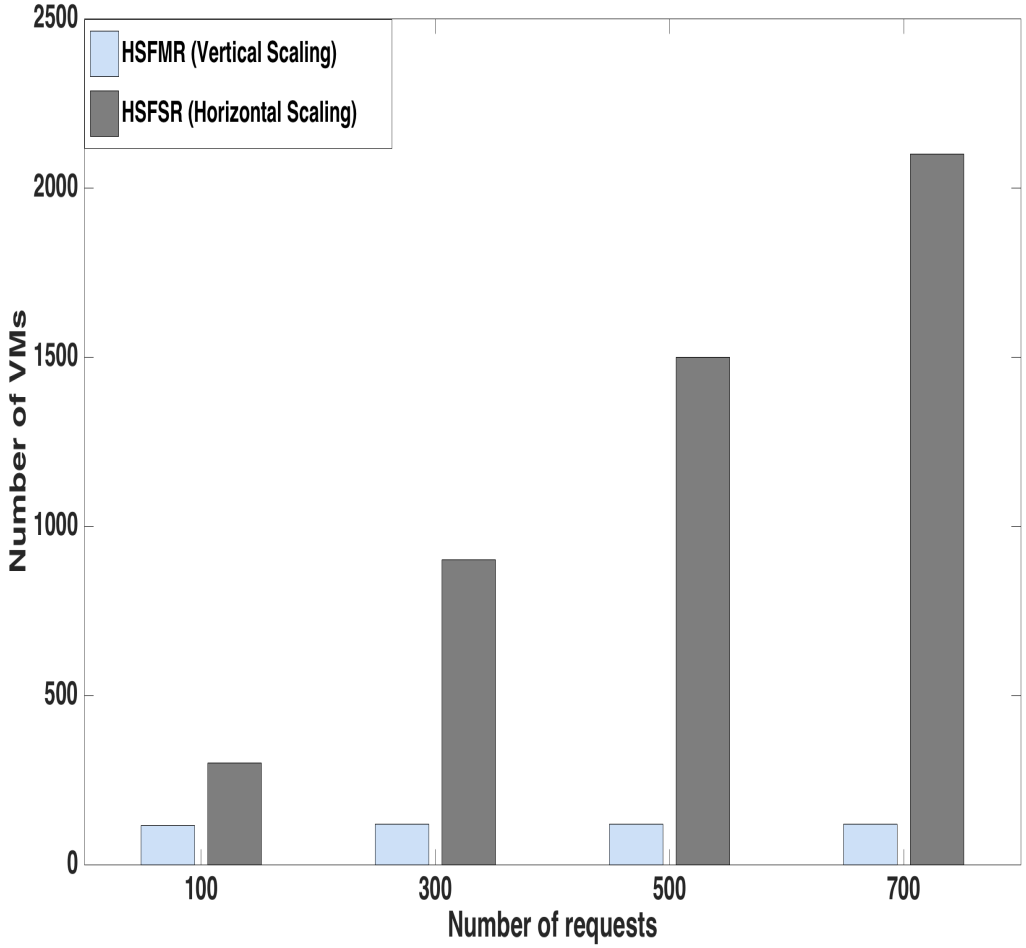

Figure 4.7: HSFSR vs. HSFMR (varying number of requests)

vestigate the impact of a number of accepted requests, on the execution time in the proposed model, we plot the average execution time for a different number of requests varying from 2 to 10. We measured the computational time for both heuristic and MILP as illustrated in Figure 4.4. The results show that the heuristic is 700 to 1000 times faster than MILP and this is the case for the considered small scale scenario. Therefore using MILP on larger network instances can be deemed as prohibited and this explains the scalability challenge we are facing.

Figure 4.5 shows that MILP results are better but close to the heuristic for a number of requests varying from 2 to 12 service chains. We note that the optimality gap is kept low and therefore from these results we can conclude that the heuristic based algorithms can find competitive solutions.

Figure 4.6 shows that our approach can decrease the average delay significantly com-

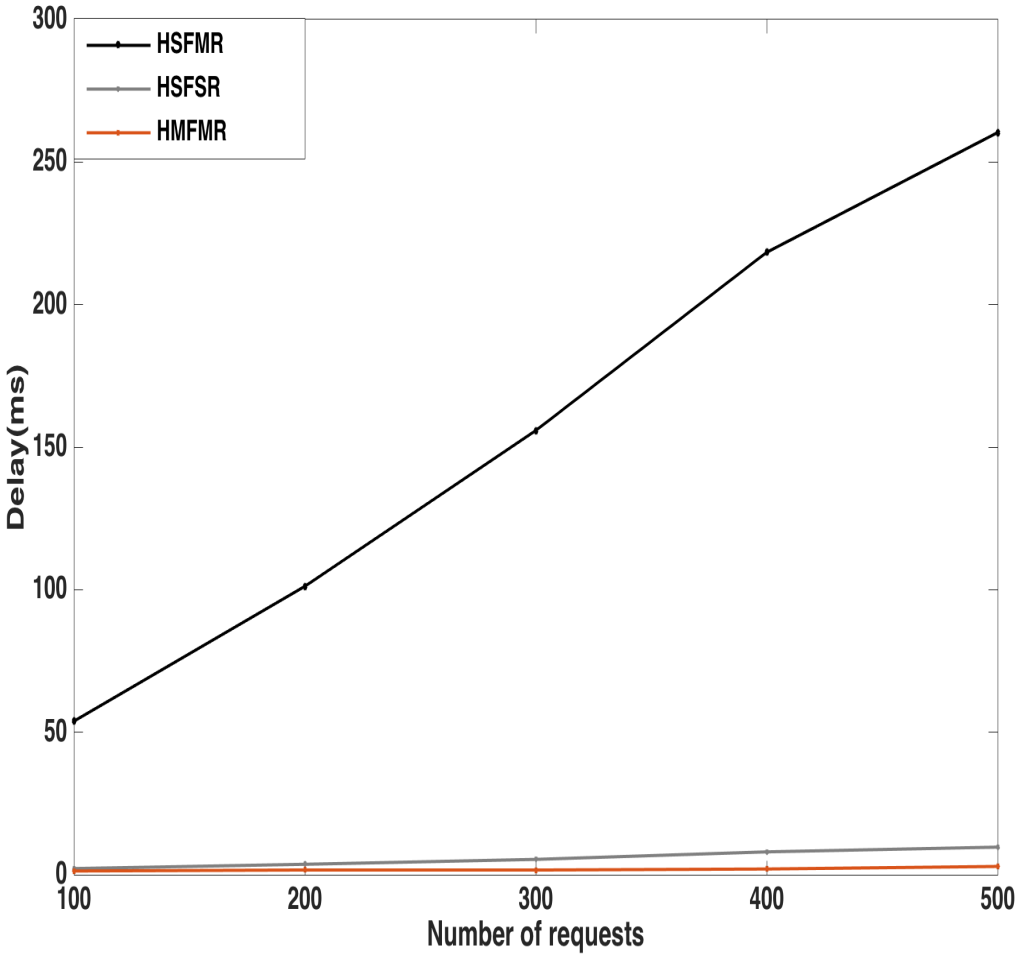

Figure 4.8: Comparing heuristics

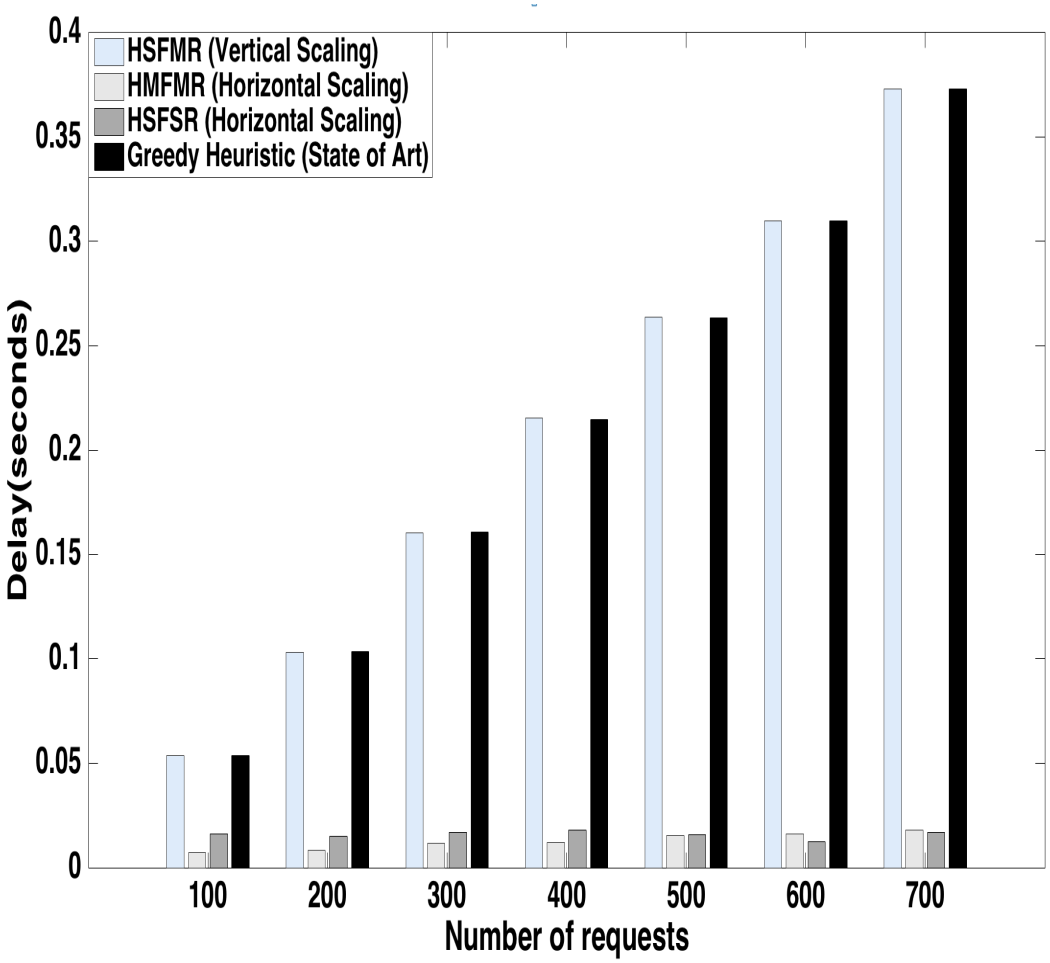

Figure 4.9: Comparison with the state of art

pared with the randomized solution.

To observe the behavior of the proposed approach, we have evaluated the heuristic methods on a large scale network scenario. The results of HSFMR show that with vertical scaling, we can decrease the number of VMs by approximately 74% compared to horizontal scaling in HSFSR. Sharing a VM instance between different chains with the same processing requirements decreases the number of VMs from 300 to 78 VMs in a random 28 edge clouds network while serving 100 requests as shown in Figure 4.7.

Figure 4.8 shows the average delay of data flows when demands vary between 100 and 500. We observe the growth of the delay measured in HSFSR, HSFMR, and HMFMR as the number of requests increases. As in the case of vertical scaling, sharing a VM instance between different requests helps to increase the capacity of the processing unit which reduces the edge cloud delay but increases the link delay.

The results from the state of the art greedy algorithm are very close to the results of those of vertical scaling and horizontal scaling but still the results of our heuristic give a better performance. Compared to the proposed state of art greedy, the delay is decreased by approximately 65% in the case of horizontal scaling and a little 1% using vertical scaling since the inter-cloud delay is the same but the link delay is different, this is shown in Figure 4.9. In our approach, we have choosen first the best shortest path before assigning the requests to the edge clouds, but in the simple greedy we first have assigned the VNFs instances of requests before routing. The first available node is loaded before going to the next one with available capacity which might cause a link bottleneck and increase the delay in that specific link. Our approach helps us to balance the load over the available resources and avoid a bottleneck.

4.7 Conclusion

In this chapter, we first formulated the VNF placement and routing problem as a non-linear integer mathematical problem and subsequently we have linearized the mathematical formulation in order to utilize powerful mixed integer mathematical solvers. The proposed approach is shown to be useful in finding optimal solutions for small to medium number of instances. This has allowed comparing the performance of a number of scale-free heuristic algorithms and validating the proposed schemes whilst evaluating the incurred delay for different models.

The results show that the MFMR approach allows us to meet a stringent latency requirement for horizontal scaling, this reduces the delay by 18% when serving 100 requests compared to HSFMR results. Scaling out VMs can provide in general a better performance in terms of delay, it also avoids us to put the machine in an off-line state to upgrade it for peak demand. Likewise, vertical scaling allows us to optimize latency, but increasing virtual resources online and dynamically might be a problem for different tenants since an interruption of the ongoing process is needed and such an interruption in the system should be planned in advance. An interesting future avenue of research is to investigate the proposed approach by considering also request admission control and VNFs affinity rules to increase the network performance and take into account potentially other network metrics.

Chapter 5

Joint Reactive and Proactive SDN Controller Assignment for Load Balancing

5.1 Introduction

The 5G technology is aiming to provide an infrastructure to ensure virtual network management and operations for an increasingly high volume of data traffic as well as supporting new services and adapt to different situations depending on the required QoS. SDN is one of the key technologies to enable a virtualized 5G system by creating, managing, and orchestrating different virtual networks built on the top of a shared infrastructure [93].

The continuous growth in applications and services based on mobile broadband systems demands operators to anticipate for a network architecture that should be able to meet future strict capacity and performance demands. Some network operators are now adopting cloud computing paradigm to cope with the tenants/users requirements, ready to match capacity demand and reduce cost by offloading communication services from dedicated hardware in operator's core to server farms located in remote data-centers. Although, the leading technologies for the next generation 5G wireless networks yet to be finalized, the essential use cases and requirements have already been listed [94],[95]. The common understanding and unanimous consensus among researchers, operators, and vendors are that the 5G will be a unified platform to deploy services such as mobile broadband, critical machine type of communications, and the IoT. The variety of use cases certainly demands a flexible new network architecture

to support a high number of different services.

NFV is the process of relocating or migrating network functions from dedicated hardware to generic servers. SDN and NFV are two closely related technologies that are often used together in a cloud computing infrastructure to have an architecture that deploys on demand "Network-as-a-Service" for users. SDN has emerged as a new intelligent architecture for network programmability. The initial idea behind SDN is to move the control-plane 'outside' the switches and enable external control of data-plane through a logical software entity called controller. Such an approach benefits mobile network management by bringing complete intelligence to the logically centralized controller. In the conventional SDN architecture, a single controller is managing all network's switches.

Considering a single controller architecture might affect the communication within the network as well as the overall performance in the case of a failure. Additionally, as requests rate increases, requests traffic sent to the controller from different network elements increases. Therefore, the capacity of the controller might not allow the processing of all requests [33]. However, a new multi-controllers SDN architecture distributes request's traffic among controllers where controllers placement can be optimized. Multi-controllers or multiple domains in SDN, as illustrated in Figure 5.1, have different domains, each with a central controller managing a group of switches. Each controller has a partial vision of the network but has to communicate with other controllers to share network information.

Having multiple controllers can overcome the problem of a single point of failure, by using other controllers when one controller is down and also, increasing the network's performance, security, and scalability. However, multi-controllers strategy raises multiple challenges such as controller placement and assignment, controllers communication cost, and load balancing.

The rest of this chapter is organized as follows. We discuss the related research work in Section 5.2. We describe the proposed QP problem, the multi-objective, the min-max models and the capacity-aware greedy algorithm in Section 5.3, where the mathematical optimization model aims to minimize the load of flow setup and make the best use of network resources. We then present the experimental setup comparing the results from the proposed model with those of the previously reported approaches in Section 5.4.

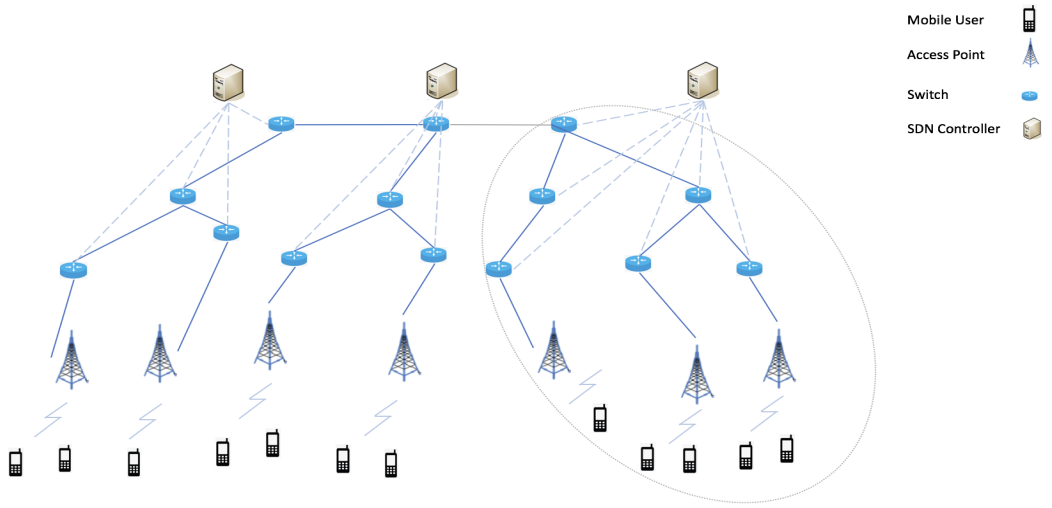

Figure 5.1: Multiple domain SDN Controllers

5.2 Background

D. Suh et al. in [96], have proposed a solution to the master controller assignment problem by minimizing the average flow set up latency in all switches along a path. One switch is connected to multiple controllers but assigned to only one master controller which is responsible for setting up flow rules. Based on the average network latency and the average arrival rate, the setup flow latency is estimated considering a number of master controllers. Among the remaining assignments that satisfy the integer non-linear programming (INLP) constraints, the authors have chosen the ones that yield to the lowest average flow set up latency.

M. J. Abdel-Rahman et al. in [97] have proposed a joint controller placement and assignment approach, and they have compared the deterministic model with a stochastic one. For this, the authors have solved the controller placement and assignment problem by minimizing the number of controllers and response time to various eNBs. Authors propose a second version, presented as the controller placement and adaptive assignment (CPPA), considering the variation of eNBs request's rates.

Y. Zhou et al. of [98] have proposed a load balancing mechanism by solving load oscillation caused by inappropriate switch migration and time efficiency problem. To do so, authors have presented a load measurement step and a switch selection group process followed by the controller selection for every preselected group of switches.

In [99], G. Wang et al. have approached the problem of multi-controllers placement

by minimizing the maximum end to end delay and controllers queuing latency. For this objective, they have started by partitioning the network into sub-networks using an updated k-means algorithm. In the second step, the authors have placed the controllers into sub-networks and assigned controllers to a cluster of switches. The results show that the algorithm achieved 2.437 times smaller latency than the performance of the k-means algorithm.

In [100], A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui have considered different metrics to assign SDN controllers to switches such as resource utilization defining the utilization rate of the processing capacity and controllers response time as the processing time to ensure the load balancing in controllers. Additionally, a one-to-many matching game based algorithm is used to solve the problem periodically. Dynamic association of the controllers to the switches shows a better performance in term of response time compared to static assignment constrained by network's load distribution.

Most of the previous work that has been detailed above have considered minimizing the latency and response time when solving the problem of controller assignment. As far as we know, none of the above works considered both reactive and proactive SDN controllers assignment considering load balancing and flow migration costs. To this end, we develop an optimization problem for the controller assignment considering both load balancing and flow migration cost. Additionally, we shed light on the inherent trade-off between load balancing and path optimization.

Table 5.1 Description of variables for mathematical models

Parameter	Domain	Description
x_{tk}	$x_{tk} \in \{0,1\}$	Equal to 1 if network element t is assigned to controller k
y_k	$y_k \in \{0,1\}$	Equal to 1 if controller k is open
C	$C \in \mathbb{N}$	Set of controllers
k	$k \in \mathbb{N}$	Defines a controller
D	$D \in \mathbb{N}$	Set of network elements
t	$t \in \mathbb{N}$	Defines a network element
l_k	$l_k \in \mathbb{R}_{>0}$	Flow traffic in controller k
r_t	$r_t \in \mathbb{R}_{>0}$	Arrival rate from t
S_k	$S_k \in \mathbb{R}_{>0}$	Processing capacity of controller k
U_k	$U_k \in \mathbb{R}_{>0}$	Utilization of controller k
K	$K \in \mathbb{N}$	Number of controllers
s_{max}	$s_{max} \in \mathbb{R}_{>0}$	Maximum total load of a controller

5.3 Problem Description and System Model

SDN separates the network control plane from the forwarding plane and via a centralized control plane. As a result, the network traffic is controlled efficiently, the management cost is reduced, and the service deployment is faster. Controllers operate as the core and the brain of an SDN domain by controlling data packets and set up flow rules. They use protocols, e.g., OpenFlow to distribute decisions such as choosing the optimal path for network traffic and overcoming performance and scalability issues.

The computational resources at the network infrastructure can be sliced and then integrated to form virtual computation platforms that can include a core network and a radio access network (RAN) to meet specific requirements. A slice can be serving a utility company, remote control for a factory or a virtual operator. The performance, requirements, and the QoS required can change from a slice to another.

In a software-defined network, all the nodes or a selection of nodes are connected to multiple controllers, where each node is managed by only one controller to be able to install flow rules. By assigning a considerable number of network's elements to the same controller, latency and other QoS requirements can easily be affected. Therefore, deploying a set of SDN controllers to manage different network's elements and network's slices is a promising approach. However, we are facing various challenges such as controllers placement and location as well as defining which controller should be assigned to which switch.

Switches send requests to controllers when they need flow rules to direct the incoming data. The assignment of switches to overloaded controllers while other controllers are underloaded can affect the response time of the requests and may result in controllers failure or losing a request. In this work, we aim to find the optimal assignment of controllers that optimize the utilization of controllers and balance the load through them considering the traffic mobility within the network. The allocation of controllers has to consider the amount of available capacity within each controller, and the traffic movement between assigned switches as well as the controller's processing capacities. Table 5.1 describes the variables and parameters we use to formulate the mathematical problem.

By considering $C = \{1, 2, ..., K\}$ the set of controllers available for assignment in the network and $D = \{1, 2, ..., T\}$ the set of switches that should be allocated to available

controllers, we define the available controllers through the binary variable y_k and the assignment of switches to controllers through the binary variable x_{tk}. Each of the network elements in which D is assigned to a single controller. Since every controller can be responsible for one or different switches, and one switch is connected to only one controller, the utilization can vary from one controller to another. As mentioned in [101], the processing delay has a significant contribution to the overall latency, and this can be affected by the controller's load when it exceeds its capacity.

5.3.1 Flow Load Balancing Cost

Load balancing is the mechanism of allocating the workload among available controllers to improve network performance. Thus, we will be minimizing the load caused by flow set up requests within the controllers to avoid over-utilization of the controllers and reduce the load cost. We define the decision variables x_{tk} and y_k that represents assignment of network switch i to controller j and that controller j is opened respectively.

$$x_{tk} = \begin{cases} 1 & \text{if element } t \text{ is assigned to controller } k \\ 0 & \text{otherwise.} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if controller } k \text{ is open} \\ 0 & \text{otherwise.} \end{cases}$$

Equation 5.1 below defines the cost function L of the load balancing optimization problem forcing the distribution of controller loads across the different available controllers, where $n \geq 2$. However, hereafter we will assume $n = 2$, r_t is the arrival rate from switch t and x_{tk} is the decision variable that defines if the switch t is assigned to controller k. The first cost is defined as follow:

$$L = \sum_{k \in \mathbf{C}} (\sum_{t \in \mathbf{D}} r_t x_{tk})^n \tag{5.1}$$

5.3.2 Migration Cost

In this section, we define the second cost defined in equation 5.2 of reactive controller assignment problem as the traffic migration cost. It can be defined as the cost of traffic moving from a switch to another one, if the switches are assigned to different

controllers. As shown in Figure 5.1, the load migration occurs when a mobile user changes its access point and moves from a domain to another one where the load increases for the related controller.

The cost of migration, M is given in equation (5.2) below, where $\pi_{tt\prime}$ is the probability that traffic is migrating from switch t to $t\prime$; note that x_{tk} is the variable that defines if the switch t is assigned to controller k and also r_t denotes the flow arrival rate.

$$M = \sum_{t \in \mathbf{D}} \sum_{t\prime \in \mathbf{D}} \sum_{k \in \mathbf{C}} \pi_{tt\prime} x_{tk} r_t (1 - x_{t\prime k}) \tag{5.2}$$

5.3.3 Multi-Objective Model

We define the problem of proactive and reactive controller assignment as a Quadratic Programming problem where the multi-objective function defined in (5.6) contains second order polynomial terms. As shown in Figure 5.2, the linear objective is invariant to the actual load. Therefore, the non-linearity of the quadratic objective penalizes more under-utilization of controllers rather than their overutilization, which prevents controller failure or at worst network crash [102].

To get the Pareto optimal solution, we use the weighted sum method by multiplying different objective functions by weighting coefficients, in our model w_1 and w_2 are the weights of the first and second cost respectively. Different objective functions have different orders of magnitudes and diverse range of values, which makes the comparison between different objectives difficult. Therefore, we use the true intervals of the objective function variations over the Pareto optimal set for normalization [103]. We calculate the Nadir and Utopia points as defined in equation (5.3), and (5.4) respectively, to get the best normalization result and bound both objectives between zero and one as expressed in (5.5).

$$z_i^N = \max_x \quad f_i(x) \tag{5.3}$$

$$z_i^U = \min_x \quad f_i(x) \tag{5.4}$$

$$0 \leq \frac{f_i(x) - z_i^U}{z_i^N - z_i^U} \leq 1 \tag{5.5}$$

Therefore, the objective function can be written as follows:

$$\min_{x,y} \quad w_1 \cdot \sum_{k \in \mathbf{C}} (\sum_{t \in \mathbf{D}} r_t x_{tk})^2 + w_2 \cdot \sum_{t \in \mathbf{D}} \sum_{t\prime \in \mathbf{D}} \sum_{k \in \mathbf{C}} \pi_{tt\prime} x_{tk} r_t (1 - x_{t\prime k}) \tag{5.6}$$

$$\text{s.t.} \quad \sum_{t \in \mathbf{D}} r_t x_{tk} \leq y_k S_k \qquad \forall k \in \mathbf{C} \tag{5.7}$$

$$\sum_{k \in \mathbf{C}} x_{tk} = 1 \qquad \forall t \in \mathbf{D} \tag{5.8}$$

$$x_{tk} \leq y_k \qquad \forall k \in \mathbf{C}, \forall t \in \mathbf{D} \tag{5.9}$$

$$\sum_{k \in \mathbf{C}} y_k \leq K \tag{5.10}$$

$$x_{tk}, y_k \in \{0, 1\} \tag{5.11}$$

In the multi-objective formulation, constraint (5.7) ensures that the capacity of controllers is respected. Constraint (5.8) forces every network element to be assigned to only one controller. Constraint (5.9) makes sure that a controller can be assigned. Constraint (5.10) makes sure that the number of opened controllers respect the predefined maximum number of controllers to use. Constraint (5.11) makes sure x_{tk}, y_k are binary.

5.3.4 Min-max Model

Due to scalability issues and to implement the proposed model within a more realistic scenario, in this section we present a min-max model. To linearize the min-max formulation, we define one integer decision variable s_{max} in equation (5.13), two binary decision variables x_{tk} and y_k defined previously in Section 5.3.3 and constraint (5.15) is added to ensure that the load of each controller $|C|$ is not exceeding s_{max} value. As defined in [104] the load of a controller $|D|$ can be defined and calculated in equation (5.12).

$$L_k = \sum_{t \in \mathbf{D}} r_t x_{tk} \tag{5.12}$$

$$s_{max} = \max \quad \sum_{t \in \mathbf{D}} r_t x_{tk} \tag{5.13}$$

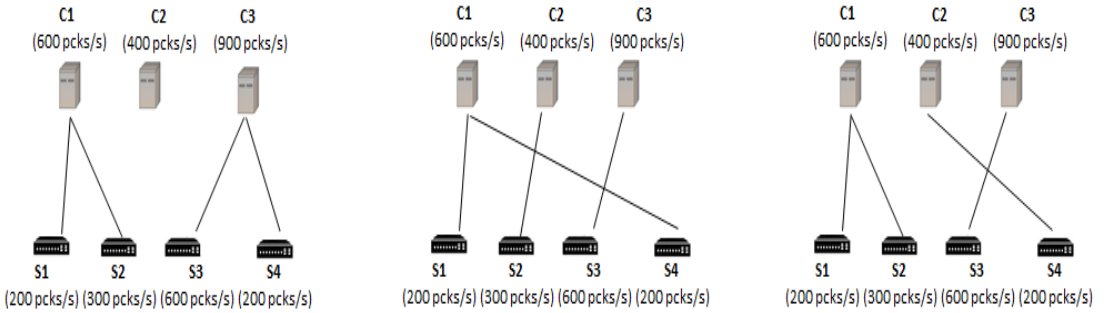

Figure 5.2: In (a), the allocation of switches to controllers is constrained by the capacity of controllers only; the cost function without the inner summation square results in the value: (200+300)+(600+200) = 1300, while with the central summation squared, the value is $(200 + 300)^2 + (600 + 200)^2 = 890000$. In (b),the optimal placement the cost function value without the inner summation square is $(200 + 200) + 300 + 600 = 1300$, while the value of the objective function when the inner summation is squared is $(200 + 200)^2 + 300^2 + 600^2 = 610000$. In (c),the assignment of controllers is one of the possible solutions to the min-max approach, the cost function value without the inner summation square is $(200+300)+200+600 = 1300$, while the value of the objective function when the inner summation is squared is $(200 + 300)^2 + 200^2 + 600^2 = 650000$.

Minimizing the maximum load is an inherently non-linear optimization problem, which can be transformed to an integer linear problem as follows:

$$\min \quad s_{max} \tag{5.14}$$

$$\text{s.t.} \quad L_k \leq s_{max} \qquad \forall k \in \mathbf{C} \tag{5.15}$$

$$L_k \leq y_k S_k \qquad \forall k \in \mathbf{C} \tag{5.16}$$

$$\sum_{k \in \mathbf{C}} x_{tk} = 1 \qquad \forall t \in \mathbf{D} \tag{5.17}$$

$$x_{tk} \leq y_k \qquad \forall k \in \mathbf{C}, \forall t \in \mathbf{D} \tag{5.18}$$

$$\sum_{k \in \mathbf{C}} y_k \leq K \tag{5.19}$$

$$x_{tk}, y_k \in \{0, 1\} \tag{5.20}$$

5.3.5 Capacity-aware Greedy Algorithm

In this section, we define a greedy algorithm for the controller assignment problem similar to [105], where the greedy heuristic is used to solve the controller placement problem. The greedy heuristic is an aware capacity algorithm and makes the assignment based on the nearest controller.

Algorithm 6 starts from a list of switches ranked by their traffic load in descending order. Then, we calculate the shortest path between switch i and the controllers, if the nearest controller j have enough capacity, switch i is assigned to controller j. In the case where controller j does not have enough resources, the algorithm chooses the next closest controller with enough capacity. The worst case will be if no controller can be assigned to a switch i due to the lake of capacity, the algorithm fails.

As presented in [106], the problem of robustness for switches assignment can be affected by the controller running out of capacity. The issue of a controller failure can be solved by replacing a failed controller or a dynamic controller reassignment, such as in [107].

5.4 Performance Evaluation

In this section, we analyze and evaluate the performance of the models presented in Section 5.3. We implement a Quadratic Programming problem using Tomlab

Algorithm 6 Capacity-aware Greedy

Input: Topology G(D x D) and list of controllers C and their placement
Output: Assignment of switches to controllers
Sort Desc Switches by traffic load
Foreach ($Switch_i$ in G)
  Calculate Distance between $switch_i$ and the $controllers_j$
  If Capacity of $controllers_j$ is sufficient
    Assign Switch $switch_i$ to the closest Controller
  Otherwise
    Assign $Switch_i$ to the next closest Controller in G
  Update Controller capacity
  EndIf
EndFor

environment in Matlab following the general form described in Appendix C.

Additionally, we analyze the results of the min-max model with the objective of minimizing the traffic load within the controllers. To evaluate the models, we compare the min-max approach, the quadratic model with K-means and capacity aware greedy algorithm under a real topology from the Internet Topology Zoo [108] which is commonly used in the literature to solve related problems [109] [110]. The topology used for the simulations is the Internet2 OS3E [111] composed of 34 switches.

Therefore, we set the parameters of the simulation as follows: the number of switches in set D(i.e.,$|D|$) to 34 switches and for each value of $|D|$, the number of possible controllers (i.e.,$|C|$) assigned to $|D|$ varies from 5 to 10 controllers. Defining the weights for different objectives can make changes in terms of both load balancing and migration cost, as shown in Figure 5.3.

Defining the weight value for each objective function can sometimes be a challenge, but the choice of the weights for each objective can be determined based on the priorities and the goals to be achieved. Figure 5.3 shows that load balancing cost is increasing as the weights on the load balancing cost increase compared to the migration cost; this shows the trade-off between load balancing and migration costs.

K-means is an algorithm used mainly to solve partitioning or clustering problems. In many previous works, the K-means algorithm is used to solve the problem of Controller Placement Problem (CPP) such as [109] and [112]. Algorithm 7 describes the K-means algorithm, where four main steps are explained.

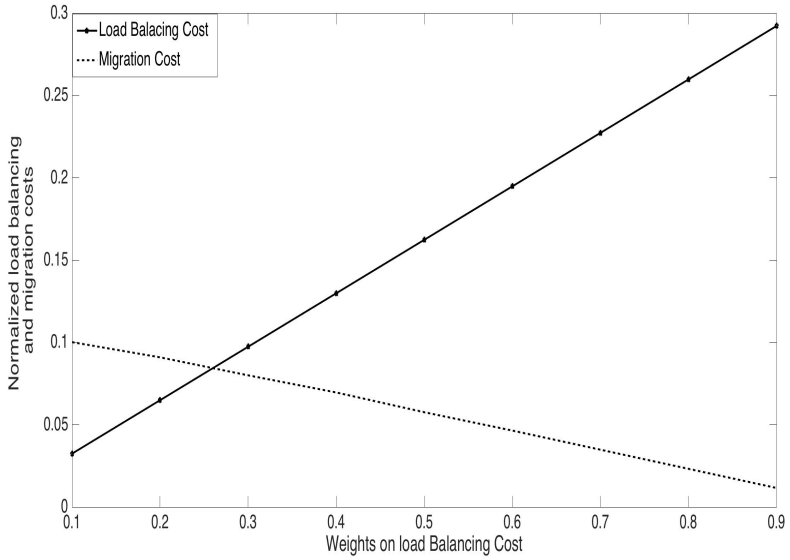

Figure 5.3: Impact of weights on different costs

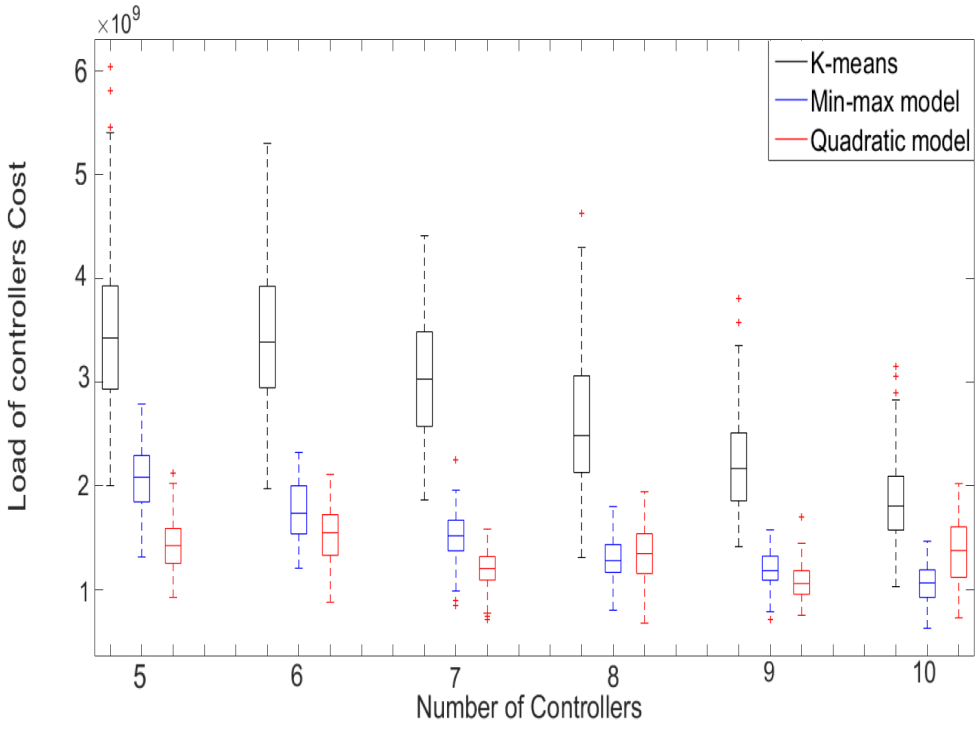

Figure 5.4: K-means algorithm Vs Min-max Model and QP model (controllers number changing)

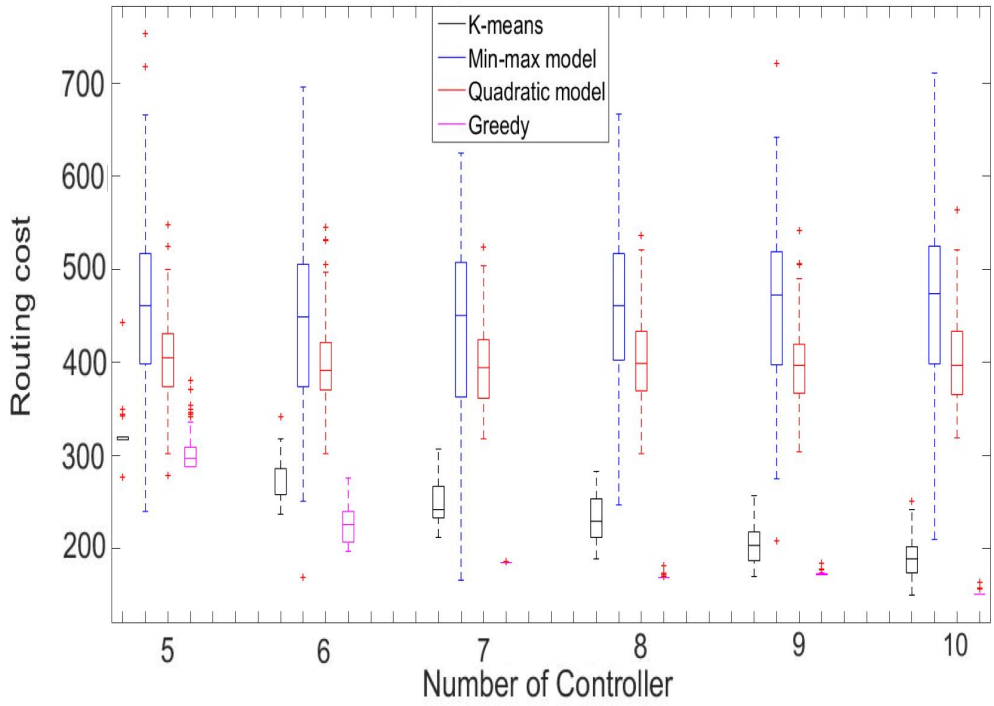

Figure 5.5: K-means algorithm, Min-max model, QP model and Greedy-based capacity algorithm (controllers number changing)

Algorithm 7 K-means Algorithm

Step 1: Initialize K centers randomly.
Step 2: Find the shortest path between nodes.
Step 3: Assign nodes to the nearest center.
Step 4: Recalculate the centers for each cluster based on the sum of distances (node with the minimum total of distances to all other nodes in the cluster).
Step 5: Repeat steps 3, 4 until no changes are made.

Table 5.2 Average load cost (.e+09)

	5	**6**	**7**	**8**	**9**	**10**
K-means Model	3.46	3.01	2.61	2.22	1.85	1.66
Min-max Model	2.07	1.76	1.52	1.30	1.18	1.05
Quadratic Model	1.43	1.53	1.19	1.35	1.07	1.36
Greedy Algorithm	18.68	24.16	30.66	35.76	39.81	44.20

To compare the performance of various schemes, we run Monte Carlo simulation with 100 iterations for different number of controllers, from 5 to 10 controllers and for several models. The objective is to measure the total cost of controllers load and the routing cost. Figure 5.4 compares the results of the calculated total load balancing cost from the proposed quadratic model, the min-max approach, and the K-means algorithm, where the number of controllers is varying from 5 to 10.

From the box plot in Figure 5.4, it is evident that for both K-means algorithm and the Min-max model, controllers load cost are decreasing as the number of controllers increases, this is because the controllers are less loaded and more capacity is available to balance the upcoming flow load from the switches through the controllers. The simulation results also indicate that the quadratic model, similar to the min-max approach, improves network load balancing compared to K-means that defines the clusters based on the shortest path and distance cost.

The greedy capacity aware algorithm is based on the shortest path; therefore, the closest controller is fully utilized before moving to the next controller. As shown in Table 5.2, there is a big gap between the average total load of controllers measured for the greedy algorithm and the rest of the models. Figures 5.4 and 5.5 show the trade-off between the load balancing cost and the path cost.

As shown in Figure 5.4 the average gain in load cost of the min-max model compared to K-means across the different number of controllers is approximately 38% for five controllers and this percentage decreases as the number of controllers increases. However, Figure 5.5 shows that the greedy and K-means perform better than the min-max and the quadratic models with an average of 22% in terms of routing cost and the gap between the results presented of the min-max and K-means models increases as the number of controllers increases.

5.5 Conclusion

For small scale networks, a single SDN controller is usually adopted. However, this might not be the case in large scale operational networks spanning a wide geographical area that requires high reliability and low latency in distributing network policies. Hence, a multi-controllers approach is more suitable to process all the requests. The multi-domains approach helps to balance and distribute the load between the controllers and decreases the time required for a switch to send a request to a controller.

In this chapter, we have proposed a mobility aware SDN-based multi-controllers load balancing approach. We have obtained an optimal controllers assignment considering both load balancing and migration cost, and we have also shown the trade-off between these two metrics. Additionally, a large set of simulation results have shown that the proposed approach can achieve a better performance in terms of load balancing between controllers than that of the standard K-means, which is widely used.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have investigated two independent but related subjects, VNFs chains placement and routing, and SDN controller assignments.

As presented in Chapter 3, we have approached the problem of chaining and routing of VNFs with the objective of delay minimization. To do so, we model the problem mathematically as a MILP problem. We have considered the cost of routing represented in a bi-objective cost function, in addition to the queuing delay cost generated by serving user/tenant requests. As a result, the evaluation is showing the trade-off between the link queuing delay and the routing cost. In order to keep the delay at the minimum, the chosen path might not be the lowest in terms of the routing cost. Additionally, the results shows that when the chosen paths have more common links, the delay increases rapidly, which increases the inflation rate of the link delay.

For a better performance, it is essential to increase the number of processed service requests, to do so we approach the problem of minimizing the overall accumulated latency in Chapter 4. We have considered the delay in both: edge clouds and links with respect to the distribution and utilization of available resources. Using MILP, we have solved the problem of VNFs placement and routing for small to medium scale networks. Also, we have proposed several heuristic algorithms and compared the performance of models for horizontal and vertical scaling using a greedy algorithm from previous research work. The results show that with a multi-features multi-requests approach we have achieved a better performance in terms of overall delay when scaling VMs horizontally compared to other methods such as the presented

greedy from the state of art and the approach based vertical scaling.

However, when using horizontal scaling, a VM is considered as a scaling unit, this can cause unnecessary over-provisioning and high resource wastage [113]. Allocating more VMs increases the total amount of resources, in addition to extra overheads created when managing the VMs dynamically. As we showed in the results, the number of VMs is three times higher using horizontal scaling compared to when using vertical scaling. Additionally, we have showed that we could optimize delay with vertical scaling, but it creates a trade-off between delay and virtual resources utilization.

In Chapter 5, we have considered a multi-controllers architecture in SDN based network in order to balance traffic load among different controllers. In this work, we focus on optimizing SDN controller assignments considering both load balancing and migration cost. A bi-objective model has been presented along with a min-max approach and a capacity-aware greedy algorithm.

However, in the bi-objective model each function is assigned to a weight. The weight values can vary based on the main goal of the optimization model, as we shown in Figure 5.3. We measure both the load cost and the routing cost. As a result, reaching a better traffic load balance between SDN controllers can result in increasing the cost of chosen paths. We also compared our approach results with K-means algorithm which results in a higher load balancing cost because they create clusters with the minimum distance cost.

The results and evaluation have shown that our proposed method could achieve a better performance in terms of load balancing compared to the standard algorithms.

6.2 Future Work

The research and contribution presented in this thesis raised different open questions. Therefore, in this section, we will be presenting various ideas and methodologies, which can improve the results obtained in this work and inspire further investigations.

- The next step for the fifth generation will be allowing dual connectivity of both 4G and 5G access network to NFV and SDN based 5G core network, which enables the implementation of network slicing. The main challenge in network slicing is the required isolation.

Therefore, network slicing is the key to offer various services with different requirements (which may be conflicting requirements) to a significant number of devices and applications simultaneously. Many questions have been raised, such as the type of resources and network function sharing that can be used among slices.

- In Chapter 5, we have presented the assignment of switches to SDN controllers. However, the allocation approach is static, and it does not re-allocate previously allocated switches to make more efficient use of available or newly added resources. Switches migration, can have a good impact when one of the controllers are down, or when there is a change in the network, which can be beneficial in avoiding any system crash. Therefore, adding switches migration will be an added value to the present work.

- Artificial intelligence and machine learning technologies have an incredible impact on telecommunication networks by improving operations and opening new opportunities. As a machine learning technique, deep learning has experienced tremendous growth and promised potential improvements in various areas.

  Therefore, it can be interesting to apply deep learning to an optimization problem as proposed in [114]. By translating the optimization problem of controller placement and assignment to a grey-scale image as input for deep convolutional neural network (CNN). Using the outputs of the optimal solutions, we train the neural network. Thus, this approach needs further investigations and can be evaluated in a real-time scenario and for a larger-scale network compared to the traditional optimization methods.

- In multi-controllers architecture for SDN based networks, at a given point of time, the traffic loads increases. Therefore, controller workloads increase but only for some time. A solution to the additional workload can be to install a new controller to handle the extra load that can not be processed by current controllers due to capacity constraint. Regardless, when the extra load can be shared between controllers with available capacities, resources can be wasted. In future work, we would like to explore the idea of load migration and switches reassignment to extend the work presented in Chapter 5. Considering switches reassignment, load migration from overloaded controllers to under-loaded ones [115] can be done dynamically in order to improve load balancing, resource utilization, and the response time.

Bibliography

[1] MD Ananth and Rinki Sharma. "Cloud Management Using Network Function Virtualization to Reduce CAPEX and OPEX". *2016 8th International Conference on Computational Intelligence and Communication Networks (CICN)*, 5(4):43–47, Dec 2016.

[2] Lemke Andreas. "6 concrete ways NFV can save you money". https://www.nokia.com/blog/6-concrete-ways-nfv-can-save-you-money. *Nokia*, Dec 2014.

[3] Massimo Condoluci and Toktam Mahmoodi. "Softwarization and virtualization in 5G mobile networks Benefits trends and challenges". *Computer Networks*, 146:65–84, Dec 2018.

[4] Saniya Zahoor and Roohie Naaz Mir. "Virtualization and IoT Resource Management: A Survey". *International Journal of Computer Networks and Applications (IJCNA)*, 5(4):43–51, 2018.

[5] Gouareb Racha, Friderikos Vasilis, and Aghvami Hamid. "Virtual Network Functions Routing and Placement for Edge Cloud Latency Minimization". *IEEE Journal on Selected Areas in Communications*, 36(10):2346–2357, Sep 2018.

[6] Gouareb Racha, Friderikos Vasilis, and Aghvami Hamid. "Delay Sensitive Virtual Network Function Placement and Routing". *25th International Conference on Telecommunications (ICT)*, pages 394–398, Jun 2018.

[7] Racha Gouareb, Vasilis Friderikos, and Hamid Aghvami. "Placement and Routing of VNFs for Horizontal Scaling". *26th International Conference on Telecommunications (ICT)*, pages 154–159, Apr 2019.

[8] Racha Gouareb, Vasilis Friderikos, Hamid Aghvami, and Mallik Tatipamula. "Joint Reactive and Proactive SDN Controller Assignment for Load Balancing". *IEEE Global Communications Conference (GLOBECOM)*, 2019.

[9] H. Ji, S. Park, J. Yeo, Y. Kim, J. Lee, and B. Shim. "Ultra-Reliable and Low-Latency Communications in 5G Downlink: Physical Layer Aspects". *IEEE Wireless Communications*, 25(3):124–130, June 2018.

[10] Olav Queseth, Ömer Bulakci, Panagiotis Spapis, Pascal Bisson, Patrick Marsch, Paul Arnold, Peter Rost, Qi Wang, Rolf Blom, Stefano Salsano, et al. "5G PPP Architecture Working Group: View on 5G Architecture". *European Commission*, Dec 2017.

[11] Ali A Zaidi, R Baldemair, M Andersson, S Faxér, V Molés-Cases, and Z Wang. "Designing for the future: the 5G NR physical layer". *Ericsson Technology Review. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/designing-for-the-future-the-5g-nr-physical-layer*, July 2017.

[12] Bob Muro. "Using a COTS SDR as a 5G Development Platform". *Microwave Journal*, 62(2), 2019.

[13] Haijun Zhang, Na Liu, Xiaoli Chu, Keping Long, Hamid Aghvami, and Victor C. M. Leung. "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges". *IEEE Communications Magazine*, 55(8):138–145, 2017.

[14] M. Bagaa, T. Taleb, and A. Ksentini. "Service-aware network function placement for efficient traffic handling in carrier cloud". *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 2402–2407, April 2014.

[15] Tarik Taleb and Adlen Ksentini. "Gateway Relocation Avoidance-aware Network Function Placement in Carrier Cloud". *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pages 341–346, 2013.

[16] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. "A comprehensive survey of network function virtualization". *Computer Networks*, 133:212–262, 2018.

[17] Rajendra Chayapathi, Syed F Hassan, and Paresh Shah. *"Network Functions Virtualization (NFV) with a Touch of SDN: Netw Fun Vir (NFV ePub_1)".* Addison-Wesley Professional, Nov 2016.

[18] Jon Tate, Pall Beck, Peter Clemens, Santiago Freitas, Jeff Gatz, Michele Girola, Jason Gmitter, Holger Mueller, Ray O'Hanlon, Veerendra Para, et al. *"IBM and Cisco: together for a world class data center".* IBM Redbooks, Jul 2013.

[19] Dejana T Vojnak, Borislav S Đorđević, Valentina V Timčenko, and Svetlana M Štrbac. "Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation". pages 1–4, 2019.

[20] S. Sultan, I. Ahmad, and T. Dimitriou. "Container Security: Issues, Challenges, and the Road Ahead". *IEEE Access*, 7:52976–52996, Apr 2019.

[21] Sonja Filiposka, Anastas Mishev, and Carlos Juiz. "Balancing performances in online VM placement". pages 153–162, 2015.

[22] Tung-Wei Kuo, Bang-Heng Liou, Kate Ching-Ju Lin, and Ming-Jer Tsai. "Deploying chains of virtual network functions: On the relation between link and server usage". *IEEE/ACM Transactions on Networking (TON)*, 26(4):1562–1576, 2018.

[23] LA Rocha and FL Verdi. "A network-aware optimization for VM placement". pages 619–625, 2015.

[24] Defang Li, Peilin Hong, Kaiping Xue, and Jianing Pei. "Virtual network function placement and resource optimization in NFV and edge computing enabled networks". *Computer Networks*, 152:12–24, 2019.

[25] Dan Li, Julong Lan, and Peng Wang. "Joint service function chain deploying and path selection for bandwidth saving and VNF reuse". *International Journal of Communication Systems*, 31(6):e3523, 2018.

[26] Aris Leivadeas, George Kesidis, Mohamed Ibnkahla, and Ioannis Lambadaris. "VNF Placement Optimization at the Edge and Cloud". *Future Internet*, 11(3):69, 2019.

[27] Hassan Hawilo, Manar Jammal, and Abdallah Shami. "Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud". *IEEE Journal on Selected Areas in Communications*, 37(3):643–655, 2019.

[28] Marco Savi, Massimo Tornatore, and Giacomo Verticale. "Impact of processing-resource sharing on the placement of chained virtual network functions". *IEEE Transactions on Cloud Computing*, 2019.

[29] Oussama Soualah, Marouen Mechtri, Chaima Ghribi, and Djamal Zeghlache. "Online and Batch Algorithms for VNFs Placement and Chaining". *Computer Networks*, 2019.

[30] Mohammad M Tajiki, Stefano Salsano, Luca Chiaraviglio, Mohammad Shojafar, and Behzad Akbari. "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining". *IEEE Transactions on Network and Service Management*, 16(1):374–388, 2019.

[31] Mohammad M Tajiki, Stefano Salsano, Mohammad Shojafar, Luca Chiaraviglio, and Behzad Akbari. "Energy-efficient path allocation heuristic for service function chaining". In *21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–8, 2018.

[32] Yong Li and Min Chen. "Software-defined network function virtualization: A survey". *IEEE Access*, 3:2542–2553, 2015.

[33] Tao Hu, Zehua Guo, Peng Yi, Thar Baker, and Julong Lan. "Multi-controller based software-defined networking: A survey". *IEEE Access*, 6:15980–15996, 2018.

[34] Hadar Sufiev, Yoram Haddad, Leonid Barenboim, and José Soler. "Dynamic SDN Controller Load Balancing". *Future Internet*, 11(3):75, 2019.

[35] Victoria Huang, Gang Chen, Qiang Fu, and Elliott Wen. "Optimizing Controller Placement for Software-Defined Networks". *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 224–232, Apr 2019.

[36] Tao Hu, Peng Yi, Zehua Guo, Julong Lan, and Yuxiang Hu. "Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks". *Future Generation Computer Systems*, 95:681–693, 2019.

[37] Ahmad Jalili, Manijeh Keshtgari, and Reza Akbari. "A new framework for reliable control placement in software-defined networks based on multi-criteria clustering approach". *Soft Computing*, pages 1–20, May 2019.

[38] Omar Alhussein, Phu Thinh Do, Junling Li, Qiang Ye, Weisen Shi, Weihua Zhuang, Xuemin Shen, Xu Li, and Jaya Rao. "Joint VNF placement and multicast traffic routing in 5G core networks". *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[39] Dale Liu. "Cisco CCNA/CCENT Exam 640-802, 640-822, 640-816 Preparation Kit". 2009.

[40] S. Waleed, M. Faizan, M. Iqbal, and M. I. Anis. "Demonstration of single link failure recovery using Bellman Ford and Dijikstra algorithm in SDN". *International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)*, pages 1–4, April 2017.

[41] N. Makariye. "Towards shortest path computation using Dijkstra algorithm". *International Conference on IoT and Application (ICIOT)*, pages 1–3, May 2017.

[42] Q. Zhang, Y. Xiao, F. Liu, J. C. S. Lui, J. Guo, and T. Wang. "Joint Optimization of Chain Placement and Request Scheduling for Network Function Virtualization". pages 731–741, June 2017.

[43] Satyam Agarwal, Francesco Malandrino, Carla Fabiana Chiasserini, and Swades De. "VNF placement and resource allocation for the support of vertical services in 5G networks". *IEEE/ACM Transactions on Networking*, 27(1):433–446, 2019.

[44] John W Chinneck. "Practical optimization: a gentle introduction". *Systems and Computer Engineering)Carleton University, Ottawa. http://www. sce. carleton. ca/faculty/chinneck/po. html*, 2006.

[45] Johannes Bisschop. *"AIMMS optimization modeling".* Lulu.com, 2006.

[46] H Paul Williams. *"Model building in mathematical programming".* John Wiley & Sons, Jan 2013.

[47] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". *Acta numerica*, 4:1–51, 1995.

[48] Shen Lin and Brian W Kernighan. "An effective heuristic algorithm for the traveling-salesman problem". *Operations research*, 21(2):498–516, 1973.

[49] Maad Mijwel. "Heuristic Algorithms". may 2015.

[50] David S. Johnson, Alan Demers, Jeffrey D. Ullman, Michael R Garey, and Ronald L. Graham. "Worst-case performance bounds for simple one-dimensional packing algorithms". *SIAM Journal on computing*, 3(4):299–325, 1974.

[51] Rashid Mijumbi, Joan Serrat, and Juan-Luis Gorricho. "Autonomic resource management in virtual networks". *arXiv preprint arXiv:1503.04576*, 2015.

[52] S. Mehraghdam, M. Keller, and H. Karl. "Specifying and placing chains of virtual network functions". *IEEE 3rd CloudNet*, pages 7–13, Oct 2014.

[53] Freddy C Chua, Julie Ward, Ying Zhang, Puneet Sharma, and Bernardo A Huberman. "Stringer: Balancing Latency and Resource Usage in Service Function Chain Provisioning". *IEEE Internet Computing*, 20(6):22–31, Nov 2016.

[54] A. Mohammadkhan, S. Ghapani, G. Liu, W. Zhang, K. K. Ramakrishnan, and T. Wood. "Virtual function placement and traffic steering in flexible and dynamic software defined networks". *The 21st IEEE International Workshop on LANMAN*, pages 1–6, April 2015.

[55] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina. "Virtual Network Function placement for resilient Service Chain provisioning". *8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 245–252, Sep. 2016.

[56] L. Qu, C. Assi, and K. Shaban. "Delay-Aware Scheduling and Resource Optimization With Network Function Virtualization". *IEEE Transactions on Communications*, 64(9):3746–3758, Sept 2016.

[57] T. M. Pham and L. M. Pham. "Load balancing using multipath routing in network functions virtualization". pages 85–90, Nov 2016.

[58] Charalampos Rotsos, Daniel King, Arsham Farshad, Jamie Bird, Lyndon Fawcett, Nektarios Georgalas, M. Gunkel, Kohei Shiomoto, Aijun Wang, Andreas Mauthe, Nicholas Race, and David Hutchison. "Network Service Orchestration Standardization: A Technology Survey". *Computer Standards Interfaces*, 54, 02 2017.

[59] Lange Stanislav, Grigorjew Alexej, Zinner Thomas, Tran-Gia Phuoc, and Jarschel Michael. "A Multi-Objective Heuristic for the Optimization of Virtual Network Function Chain Placement". *2017 IEEE ITC*.

[60] R Timothy Marler and Jasbir S Arora. "Survey of multi-objective optimization methods for engineering". *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[61] Mathieu Bouet, Jérémie Leguay, Théo Combe, and Vania Conan. "Cost-based placement of vDPI functions in NFV infrastructures". *International Journal of Network Management*, 25(6):490–506, 2015.

[62] B. Fortz and M. Thorup. "Internet traffic engineering by optimizing OSPF weights". *Proceedings IEEE INFOCOM 2000.*, 2:519–528, 2000.

[63] NFVISG ETSI. "GS NFV 002-V1. 1.1-Network Function Virtualisation (NFV)-Architectural Framework". *publishing October*, 2013.

[64] "Google Virtual Machine Instances". *https://cloud.google.com/compute/docs/instances/*.

[65] "Amazon Virtual Machine Instances". *https://aws.amazon.com/ec2/instance-types/*.

[66] Deval Bhamare, Mohammed Samaka, Aiman Erbad, Raj Jain, Lav Gupta, and H. Anthony Chan. "Optimal virtual network function placement in multi-cloud service function chaining architecture". *Computer Communications*, 102:1–16, 2017.

[67] A. Laghrissi, T. Taleb, M. Bagaa, and H. Flinck. "Towards Edge Slicing: VNF Placement Algorithms for a Dynamic amp;amp; Realistic Edge Cloud Environment". pages 1–6, Dec 2017.

[68] N. Kiji, T. Sato, R. Shinkuma, and E. Oki. "Virtual Network Function Placement and Routing Model for Multicast Service Chaining Based on Merging Multiple Service Paths". *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, May 2019.

[69] T. Taleb, M. Bagaa, and A. Ksentini. "User mobility-aware Virtual Network Function placement for Virtual 5G Network Infrastructure". *IEEE International Conference on Communications (ICC)*, pages 3879–3884, June 2015.

[70] M. Bagaa, T. Taleb, A. Laghrissi, A. Ksentini, and H. Flinck. "Coalitional Game for the Creation of Efficient Virtual Core Network Slices in 5G Mobile Systems". *IEEE Journal on Selected Areas in Communications*, 36(3):469–484, March 2018.

[71] Deval Bhamare, Mohammed Samaka, Aiman Erbad, Raj Jain, Lav Gupta, and H. Anthony Chan. "Optimal Virtual Network Function Placement in Multi-cloud Service Function Chaining Architecture". *Comput. Commun.*, 102:1–16, Apr 2017.

[72] Mohammad Mahdi Tajiki, Stefano Salsano, Mohammad Shojafar, Luca Chiaraviglio, and Behzad Akbari. "Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining". *IEEE Transactions on Network and Service Management*, 16(1):374–388, March 2019.

[73] L. Fu, X. Fu, Z. Zhang, Z. Xu, X. Wu, X. Wang, and S. Lu. "Joint Optimization of Multicast Energy in Delay-Constrained Mobile Wireless Networks". *IEEE/ACM Transactions on Networking*, 26(1):633–646, Feb 2018.

[74] J. Bi, Z. Zhu, R. Tian, and Q. Wang. "Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center". *IEEE 3rd International Conference on Cloud Computing*, pages 370–377, July 2010.

[75] T. Taleb, M. Corici, C. Parada, A. Jamakovic, S. Ruffino, G. Karagiannis, and T. Magedanz. "EASE: EPC as a service to ease mobile core network deployment over cloud". *IEEE Network*, 29(2):78–88, March 2015.

[76] Zoltán Ádám Mann. "Allocation of Virtual Machines in Cloud Data Centers&Mdash;A Survey of Problem Models and Optimization Algorithms". *ACM Comput. Surv.*, 48(1):11:1–11:34, August 2015.

[77] J. Prados-Garzon, P. Ameigeiras, J. J. Ramos-Munoz, P. Andres-Maldonado, and J. M. Lopez-Soler. "Analytical modeling for Virtualized Network Functions". pages 979–985, May 2017.

[78] W. Rankothge, F. Le, A. Russo, and J. Lobo. "Optimizing Resource Allocation for Virtualized Network Functions in a Cloud Center Using Genetic Algorithms". *IEEE Transactions on Network and Service Management*, 14(2):343–356, June 2017.

[79] Aris Leivadeas, Matthias Falkner, Ioannis Lambadaris, and George Kesidis. "Optimal virtualized network function allocation for an SDN enabled cloud". *Computer Standards & Interfaces*, 54:266–278, 2017.

[80] Mohammad Mahdi Tajiki, Stefano Salsano, Mohammad Shojafar, Luca Chiaraviglio, and Behzad Akbari. "Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining". *IEEE Transactions on Network and Service Management*, 16(1):374–388, March 2019.

[81] Jin Y Yen. "Finding the k shortest loopless paths in a network". *management Science*, 17(11):712–716, 1971.

[82] N. Tsikoudis, A. Papadogiannakis, and E. P. Markatos. "LEoNIDS: A Low-Latency and Energy-Efficient Network-Level Intrusion Detection System". *IEEE Transactions on Emerging Topics in Computing*, 4(1):142–155, Jan 2016.

[83] G. Huang, S. Wang, M. Zhang, Y. Li, Z. Qian, Y. Chen, and S. Zhang. "Auto scaling virtual machines for web applications with queueing theory". *3rd International Conference on Systems and Informatics (ICSAI)*, pages 433–438, Nov 2016.

[84] James F. Kurose and Keith W. Ross. *"Computer Networking: A Top-Down Approach (6th Edition)"*. Pearson, 6th edition, 2012.

[85] Dimitri Bertsekas and Robert Gallager. *"Data Networks (2Nd Ed.)"*. Prentice-Hall, Inc., 1992.

[86] H. Zhu and C. Huang. "Cost-Efficient VNF Placement Strategy for IoT Networks with Availability Assurance". *IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, Sept 2017.

[87] N. Bouten, R. Mijumbi, J. Serrat, J. Famaey, S. LatrÃ©, and F. De Turck. "Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests". *IEEE Transactions on Network and Service Management*, 14(2):317–331, June 2017.

[88] Marcelo Caggiani Luizelli, Danny Raz, and Yaniv Sa'ar. "Optimizing NFV Chain Deployment Through Minimizing the Cost of Virtual Switching". *IEEE INFOCOM Conference on Computer Communications*, pages 2150–2158, 2018.

[89] P. Key, L. Massoulie, and D. Towsley. "Combining Multipath Routing and Congestion Control for Robustness". *40th Annual Conference on Information Sciences and Systems*, pages 345–350, March 2006.

[90] P Erdős and A Rényi. "On random graphs". *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[91] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary. "Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions". *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 98–106, May 2015.

[92] László Lovász et al. "Random walks on graphs: A survey". *Combinatorics, Paul erdos is eight 2.1*, 1993.

[93] X. Li, M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain. "Network Slicing for 5G: Challenges and Opportunities". *IEEE Internet Computing*, 21(5):20–27, 2017.

[94] N. Alliance. "5g white paper". available: https://www.ngmn.org/uploads/ media/ngmn 5g white paper v1 0.pdf. 2015.

[95] G. T. . Draft. "Feasibility study on new services and markets technology enablers, stage 1 (release 14)". [online]. available: http://www.3gpp.org/ftp/specs/archive/ 22 series/22.891/. Nov 2015.

[96] D. Suh and S. Pack. "Low-Complexity Master Controller Assignment in Distributed SDN Controller Environments". *IEEE Communications Letters*, 22(3):490–493, March 2018.

[97] M. J. Abdel-Rahman, E. A. Mazied, K. Teague, A. B. MacKenzie, and S. F. Midkiff. "Robust Controller Placement and Assignment in Software-Defined Cellular Networks". *26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9, July 2017.

[98] Y. Zhou, Y. Wang, J. Yu, J. Ba, and S. Zhang. "Load balancing for multiple controllers in SDN based on switches group". *19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 227–230, Sept 2017.

[99] G. Wang, Y. Zhao, J. Huang, and Y. Wu. "An Effective Approach to Controller Placement in Software Defined Wide Area Networks". *IEEE Transactions on Network and Service Management*, 15(1):344–355, March 2018.

[100] A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui. "SDN Controller Assignment and Load Balancing with Minimum Quota of Processing Capacity". *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2018.

[101] G. Wang, Y. Zhao, J. Huang, and W. Wang. "The Controller Placement Problem in Software Defined Networking: A Survey". *IEEE Network*, 31(5):21–27, 2017.

[102] Tao Hu, Peng Yi, Zehua Guo, Julong Lan, and Yuxiang Hu. "Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks". *Future Generation Computer Systems*, 95:681–693, 2019.

[103] Oleg Grodzevich and Oleksandr Romanko. "Normalization and other topics in multi-objective optimization". 2006.

[104] Peiying Tao, Chun Ying, Zhe Sun, Shuhua Tan, Pan Wang, and Sun Zhixin. "The Controller Placement of Software-Defined Networks Based on Minimum Delay and Load Balancing". *IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 310–313, Aug 2018.

[105] K. S. Sahoo, B. Sahoo, R. Dash, and N. Jena. "Optimal controller selection in Software Defined Network using a greedy-SA algorithm". *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 2342–2346, March 2016.

[106] Tao Hu, Peng Yi, Zehua Guo, Julong Lan, and Yuxiang Hu. "Dynamic slave controller assignment for enhancing control plane robustness in software-defined networks". *Future Generation Computer Systems*, 95:681–693, 2019.

[107] F. Acan, G. Gur, and F. Alagoz. "Reactive Controller Assignment for Failure Resilience in Software Defined Networks". *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–6, Sep. 2019.

[108] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. "The internet topology zoo". *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[109] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li. "A K-means-based network partition algorithm for controller placement in software defined network". *IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[110] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann. "Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks". *IEEE Transactions on Network and Service Management*, 12(1):4–17, March 2015.

[111] Jianxin Liao, Haifeng Sun, Jingyu Wang, Qi Qi, Kai Li, and Tonghong Li. "Density cluster based approach for controller placement problem in large-scale software defined networkings". *Computer Networks*, 112:24–35, 2017.

[112] H. Kuang, Y. Qiu, R. Li, and X. Liu. "A Hierarchical K-Means Algorithm for Controller Placement in SDN-Based WAN Architecture". *10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 263–267, Feb 2018.

[113] Marian Turowski and Alexander Lenk. "Vertical Scaling Capability of OpenStack - Survey of Guest Operating Systems, Hypervisors, and the Cloud Management Platform". *ICSOC Workshops*, 2014.

[114] Yantong Wang and Vasilis Friderikos. "Caching as an Image Characterization Problem using Deep Convolutional Neural Networks". *arXiv preprint arXiv:1907.07263*, Jul 2019.

[115] Kshira Sagar Sahoo and Bibhudatta Sahoo. "CAMD: a switch migration based load balancing framework for software defined networks". *IET Networks*, 2019.

[116] Kenneth Holmström, Anders O Göran, and Marcus M Edvall. "Users guide for tomlab/sol"., 2008.

Appendix A

Greedy Algorithm

A.1 Dijkstra's algorithm

Algorithm 8 Dijkstra's algorithm

Input Graph G, Source s
for each vertex $v \in V_G$
    $dist[v] \leftarrow \infty$
    $parent[v] \leftarrow NIL$
$dist[s] \leftarrow 0$
$Q \leftarrow V - G$
While $Q \neq \emptyset$
    $u \leftarrow ExtractMin(Q)$
    for each edge $e = (u, v)$
        if $dist[v] > dist[u] + weight[e]$
            $dist[v] \leftarrow dist[u] + weight[e]$
            $parent[v] \leftarrow u$
$H \leftarrow (V_G, \emptyset)$
for each vertex $v \in V_G, v \neq s$
    $E_H \leftarrow E_H \bigcup (parent[v], v)$
return H, dist

A.2 Bin-packing problem (Best fit decreasing algorithm)

Algorithm 9 Best-fit decreasing algorithm

Input
Bins as $B_1, B_2, \ldots, B_m$ filled to level zero
Items to be placed are in order $A = (a_1, a_2, \ldots, a_n)$

Arrange A in decreasing order
While Not all A are placed
  Find the best B_j with level $l \leq 1 - a_i$
    Place a_i in B_j
    Update B_j level to $\beta = l + a_i$

Appendix B

Yen's k-Shortest Path Algorithm (1971)

Yen's algorithm has been modelled to find k-shortest loopless paths, where any shortest path algorithm can be used to find the best path. In this thesis, the shortest path algorithm that have been used is Dijkstra algorithm. There is two steps to follow to find K-shortest paths. The first step is to find the first shortest path and the second step is to find the rest of K-1 shortest paths. Final K-shortest paths are stored in matrix A, while the potential shortest paths are stored in matrix B. Dijkstra algorithm is used to find the first shortest path, stored in $A[0]$. In order to find the shortest paths $A[k]$, where k is between 1 and $(K-1)$, we need to find all potential paths and choose the path with the minimum length.

The first process of finding the potential shortest paths follows three steps. The first step is to choose the root path $P[k]$, which is the sub-path of $A[k-1]$ from the first

node i in $A[k-1]$. The second step is to find the spur path $S[k]$ which is the shortest path from the node i to the destination node. To ensure that the spur path is unique, we remove the edge from node i to node $i+1$ by setting its cost to infinity. The summation of the root path and the spur path is added to B as a potential shortest path. The edge costs set previously to infinity are restored. The second process is to choose the path in B with the lowest cost, which is removed from B and added to A.

Algorithm 10 Yen's Algorithm

```
function YenKSP(Graph, source, destination, K):
A[0] = Dijkstra(Graph, source, destination);
B = [];

for k from 1 to K
    for i from 0 to size(A[k − 1]) − 2
      spurNode = A[k − 1].node(i);
      rootPath = A[k − 1].nodes(0, i);

      for each path p in A
          if rootPath == p.nodes(0, i)
            remove p.edge(i,i + 1) from Graph;

      for each node rootPathNode in rootPath except spurNode
        remove rootPathNode from Graph;

      spurPath = Dijkstra(Graph, spurNode, destination);
      totalPath = rootPath + spurPath;
      B.append(totalPath);
      restore edges to Graph;
      restore nodes in rootPath to Graph;

    if B is empty
        break;

    B.sort();
    A[k] = B[0];
    B.pop();
return A
```

Appendix C

Tomlab

C.1 Dijkstra's algorithm

$$\min_{x,y} \sum_{k \in \mathbf{C}} (\sum_{t \in \mathbf{D}} r_t x_{tk})^2 + \sum_{t \in \mathbf{D}} \sum_{t\prime \in \mathbf{D}} \sum_{k \in \mathbf{C}} \pi_{tt\prime} x_{tk} r_t (1 - x_{t\prime k}) \tag{C.1}$$

$$\min_{x} \quad 0.5 x^T F x + c^T x \tag{C.2}$$

$$\text{s.t} \qquad Ax \leq b \tag{C.3}$$

The general form in TOMLAB [116] for a quadratic programming problem is defined in equations (C.2) and (C.3) where x^T denotes the vector transpose of x, c a n dimensional vector and F a $n * n$ dimensional real symmetric matrix.

Therefore, we write equation (C.1) in the form of (C.2) and (C.3) to be able to use the Tomlab environment and solve the non-linear problem as a Quadratic programming problem in Chapter 5.